MPI Parallel Connected Component Counting on Overlap Graphs and ER Graphs

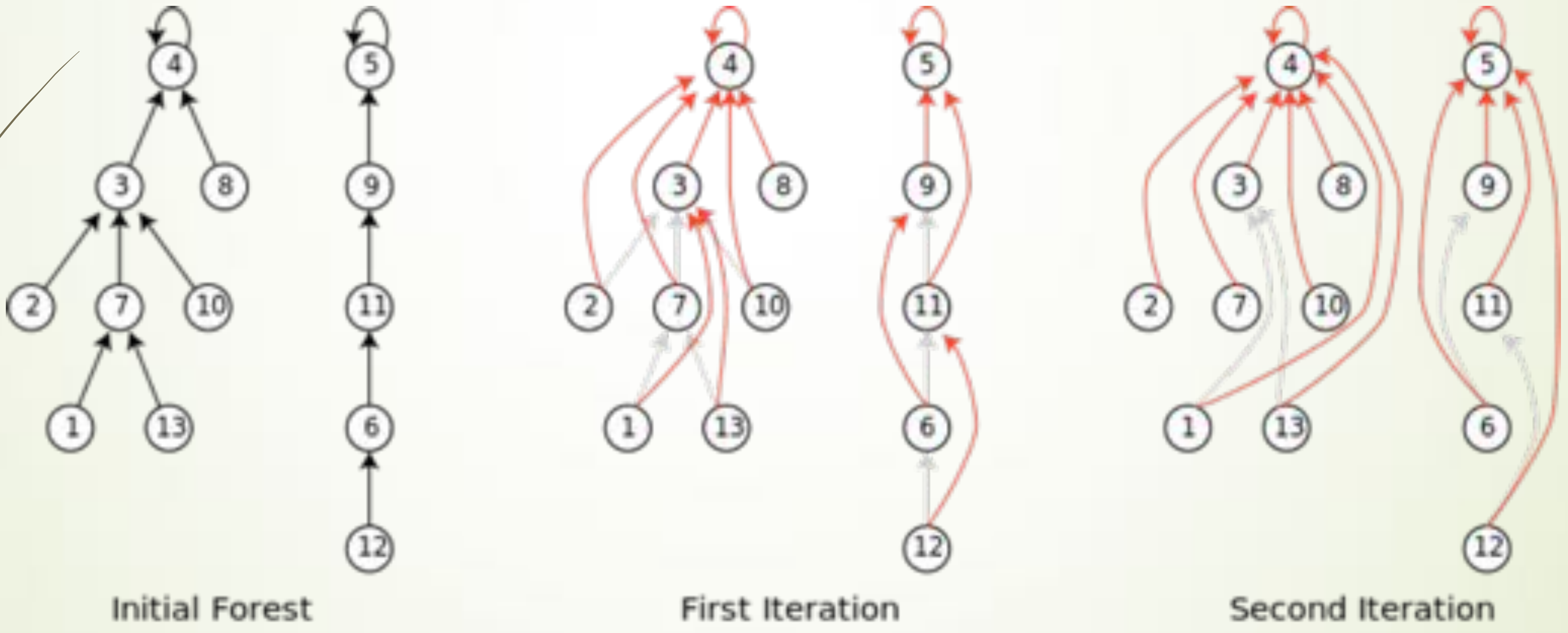
Vicky Zheng

Dr. Russ Miller

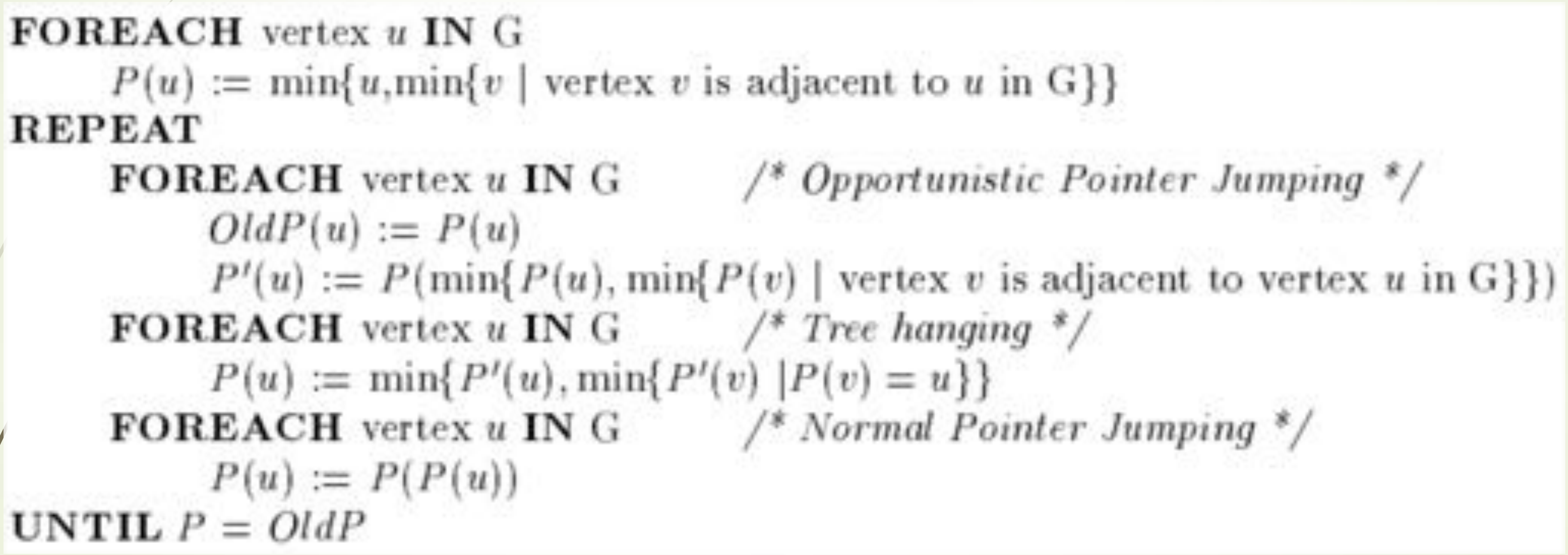
CSE 633

How to calculate number of connected components

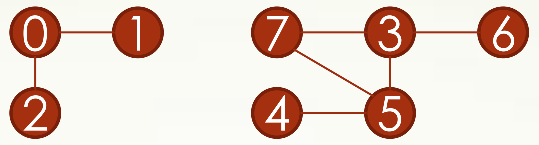
- Kumar, S., S. Goddard, and J. Prins. *Connected components algorithms for mesh-connected parallel computers*. AMS, 1997.



Algorithm



```
**FOREACH** vertex $u$ **IN** G
    $P(u) := \min\{u, \min\{v \mid \text{vertex } v \text{ is adjacent to } u \text{ in G}\}\}$
**REPEAT**
    **FOREACH** vertex $u$ **IN** G          /* Opportunistic Pointer Jumping */
        $OldP(u) := P(u)$
        $P'(u) := P(\min\{P(u), \min\{P(v) \mid \text{vertex } v \text{ is adjacent to vertex } u \text{ in G}\}\})$
    **FOREACH** vertex $u$ **IN** G          /* Tree hanging */
        $P(u) := \min\{P'(u), \min\{P'(v) \mid P(v) = u\}\}$
    **FOREACH** vertex $u$ **IN** G          /* Normal Pointer Jumping */
        $P(u) := P(P(u))$
**UNTIL** $P = OldP$
```

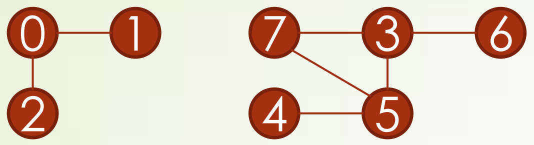


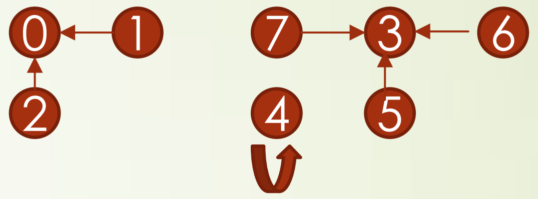
Initialization

	0	1	2	3	4	5	6	7
0	1	1	1					
1	1	1						
2	1		1					
3				1		1	1	1
4					1	1		
5				1	1	1		1
6				1			1	
7				1		1		1

$\xrightarrow{\text{Initialization}}$

	0	1	2	3	4	5	6	7
0	0	0	0					
1	1	1						
2	2		2					
3				3		3	3	3
4					4	4		
5				5	5	5		5
6				6			6	
7				7		7		7





Initialization

	0	1	2	3	4	5	6	7
0	1	1	1					
1	1	1						
2	1		1					
3				1		1	1	1
4					1	1		
5				1	1	1		1
6				1			1	
7				1		1		1

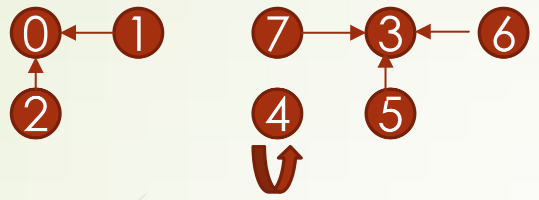
$\xrightarrow{\text{Initialization}}$

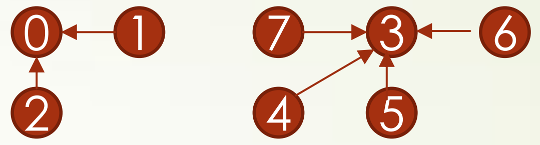
	0	1	2	3	4	5	6	7
0	0	0	0					
1	1	1						
2	2		2					
3				3		3	3	3
4					4	4		
5				5	5	5		5
6				6			6	
7				7		7		7

MPI_ALLreduce
Column Wise
MPI_MIN

$\longrightarrow$

	0	1	2	3	4	5	6	7
0	0	0	0					
1	0	0						
2	0		0					
3				3		3	3	3
4					4	3		
5				3	4	3		3
6				3			3	
7				3		3		3





Repeat until convergence

	0	1	2	3	4	5	6	7
0	0	0	0					
1	0	0						
2	0		0					
3				3		3	3	3
4					4	3		
5				3	4	3		3
6				3			3	
7				3		3		3

MPI_ALLreduce
Row - wise
MPI_MIN

$\longrightarrow$

MPI_ALLreduce
Column Wise
MPI_MIN

$\longleftarrow$

	0	1	2	3	4	5	6	7
0	0	0	0					
1	0	0						
2	0		0					
3				3		3	3	3
4					3	3		
5				3	3	3		3
6				3			3	
7				3		3		3

Data Set

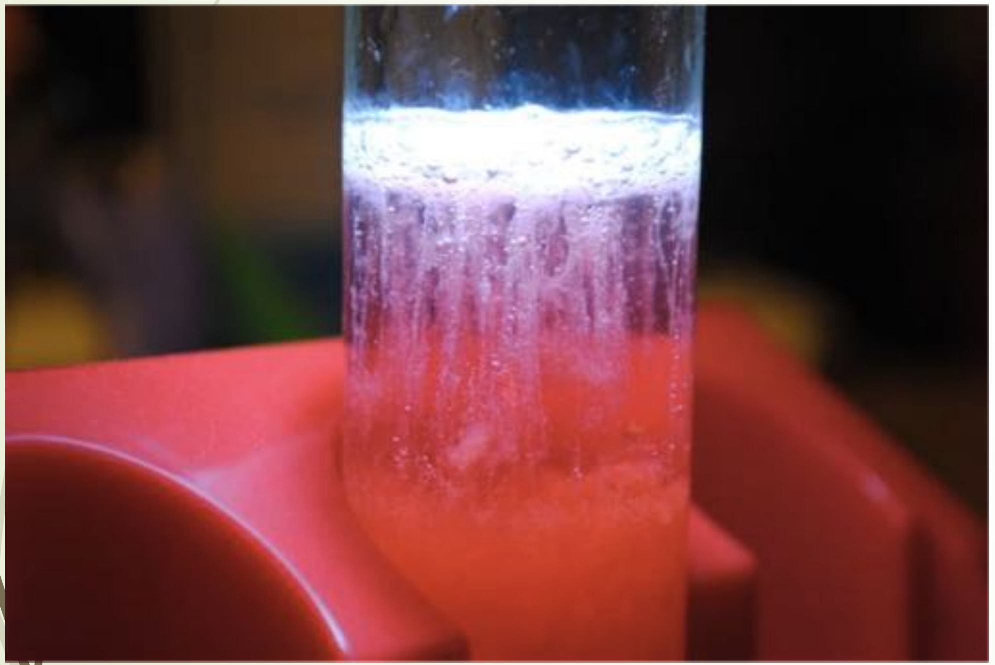
- Overlap graph of four species: Bacteroides vulgatus, Klebsiella pneumoniae, Moraxella osloensis, Streptococcus suis
- I was suppose to have 20 species in total, but the other samples were low quality (this will be explained later).
- Due to a lack of data for overlap graphs, I began using Erdős–Rényi (ER) graphs where the parameters are number of nodes and edge probability

How DNA assembly is done (recap)

- Pick and extract a sample

How DNA assembly is done (recap)

- Isolate DNA and prepare for sequencing (this is done through wet lab)

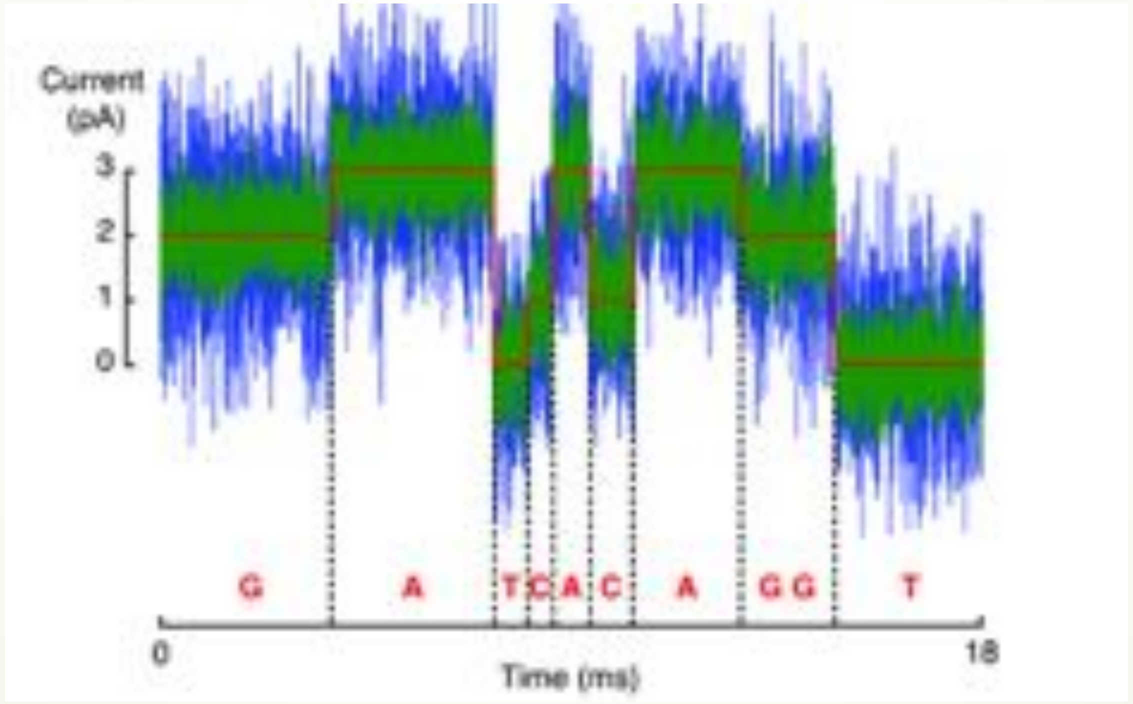


How DNA assembly is done

- Put DNA through sequencer

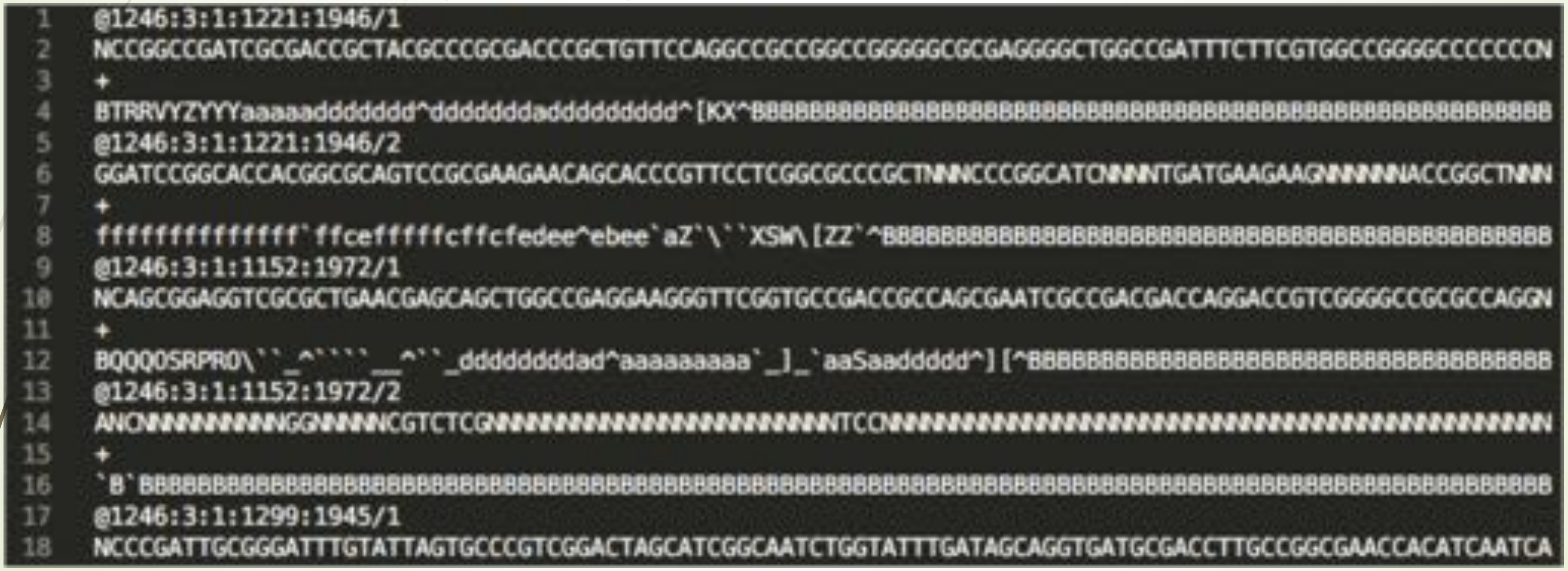
How DNA assembly is done

- Perform base calling to extract nucleotides.



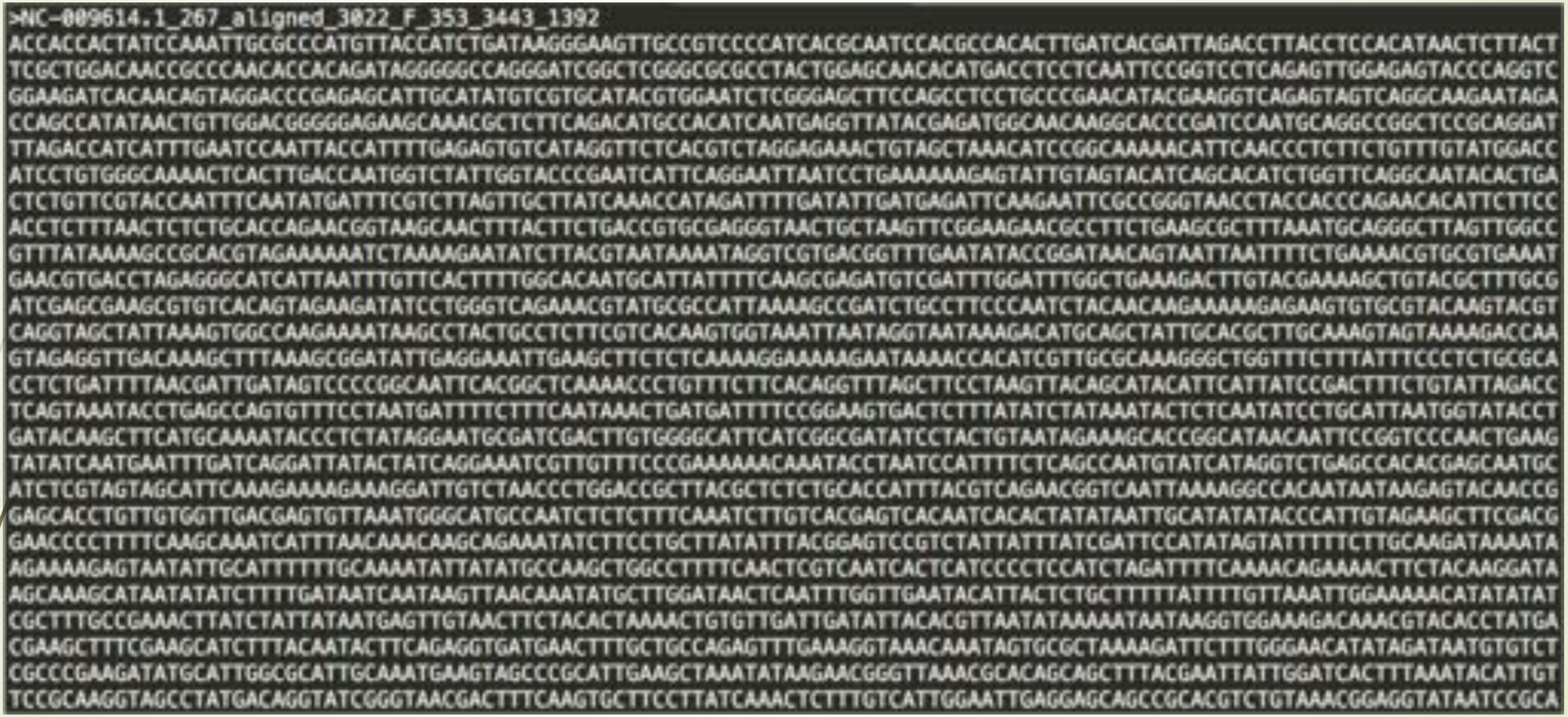
How DNA assembly is done

- Finally, you have your reads!



```
@1246:3:1:1221:1946/1
NCCGGCCGATCGCGACCGCTACGCCCGCGACCCGCTGTTCCAGGCCGCCGGCCGGGGGCGCGAGGGGCTGGCCGATTTCTTCGTGGCCGGGGCCCCCCCN
+
BTRRVYZYYYaaaaadddddd^dddddddaddddddddd^[KX^BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
@1246:3:1:1221:1946/2
GGATCCGGCACCACGGCGCAGTCCGCGAAGAACAGCACCCGTTCCTCGGCGCCCGCTNNNCCCGGCATCNNNNTGATGAAGAAGNNNNNNACCGGCTNNN
+
ffffffffffffff`ffceffffffcffcfedee^ebee`aZ`\``XSW\[ZZ`^BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
@1246:3:1:1152:1972/1
NCAGCGGAGGTCGCGCTGAACGAGCAGCTGGCCGAGGAAGGGTTCGGTGCCGACCGCCAGCGAATCGCCGACGACCAGGACCGTCGGGGCCGCGCCAGGN
+
BQQQOSRPRO\``_^````__^``_ddddddddad^aaaaaaaaa`_]_`aaSaadddd^][^BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
@1246:3:1:1152:1972/2
ANCNNNNNNNNNNGGNNNNNCGTCTCGNNNNNNNNNNNNNNNNNNNNNNNNNNNTCCNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
+
`B`BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
@1246:3:1:1299:1945/1
NCCCGATTGCGGGATTTGTATTAGTGCCCGTCGGACTAGCATCGGCAATCTGGTATTTGATAGCAGGTGATGCGACCTTGCCGGCGAACCACATCAATCA
```

After some data cleaning..



```
>NC-009614.1_267_aligned_3022_F_353_3443_1392
ACCACCACTATCCAAATTGCGCCCATGTTACCATCTGATAAGGGAAGTTGCCGTCCCCATCACGCAATCCACGCCACACTTGATCACGATTAGACCTTACCTCCACATAACTCTTACT
TCGCTGGACAACCGCCCAACACCACAGATAGGGGGCCAGGGATCGGCTCGGGCGCGCCTACTGGAGCAACACATGACCTCCTCAATTCCGGTCCTCAGAGTTGGAGAGTACCCAGGTC
GGAAGATCACAACAGTAGGACCCGAGAGCATTGCATATGTCGTGCATACGTGGAATCTCGGGAGCTTCCAGCCTCCTGCCCGAACATACGAAGGTCAGAGTAGTCAGGCAAGAATAGA
CCAGCCATATAACTGTTGGACGGGGGAGAAGCAAACGCTCTTCAGACATGCCACATCAATGAGGTTATACGAGATGGCAACAAGGCACCCGATCCAATGCAGGCCGGCTCCGCAGGAT
TTAGACCATCATTTGAATCCAATTACCATTTTGAGAGTGTCATAGGTTCTCACGTCTAGGAGAAACTGTAGCTAAACATCCGGCAAAAACATTCAACCCTCTTCTGTTTGTATGGACC
ATCCTGTGGGCAAAACTCACTTGACCAATGGTCTATTGGTACCCGAATCATTCAGGAATTAATCCTGAAAAAAGAGTATTGTAGTACATCAGCACATCTGGTTCAGGCAATACACTGA
CTCTGTTCGTACCAATTTCAATATGATTTCGTCTTAGTTGCTTATCAAACCATAGATTTTGATATTGATGAGATTCAAGAATTCGCCGGGTAACCTACCACCCAGAACACATTCTTCC
ACCTCTTTAACTCTCTGCACCAGAACGGTAAGCAACTTTACTTCTGACCGTGCGAGGGTAACTGCTAAGTTCGGAAGAACGCCTTCTGAAGCGCTTTAAATGCAGGGCTTAGTTGGCC
GTTTATAAAAGCCGCACGTAGAAAAAATCTAAAAGAATATCTTACGTAATAAAATAGGTCGTGACGGTTTGAATATACCGGATAACAGTAATTAATTTTCTGAAAACGTGCGTGAAAT
GAACGTGACCTAGAGGGCATCATTAATTTGTTCACTTTTGGCACAATGCATTATTTTCAAGCGAGATGTCGATTTGGATTTGGCTGAAAGACTTGTACGAAAAGCTGTACGCTTTGCG
ATCGAGCGAAGCGTGTCACAGTAGAAGATATCCTGGGTCAGAAACGTATGCGCCATTAAAAGCCGATCTGCCTTCCCAATCTACAACAAGAAAAAGAGAAGTGTGCGTACAAGTACGT
CAGGTAGCTATTAAAGTGGCCAAGAAAATAAGCCTACTGCCTCTTCGTCACAAGTGGTAAATTAATAGGTAATAAAGACATGCAGCTATTGCACGCTTGCAAAGTAGTAAAAGACCAA
GTAGAGGTTGACAAAGCTTTAAAGCGGATATTGAGGAAATTGAAGCTTCTCTCAAAAGGAAAAAGAATAAAACCACATCGTTGCGCAAAGGGCTGGTTTCTTTATTTCCCTCTGCGCA
CCTCTGATTTTAACGATTGATAGTCCCCGGCAATTCACGGCTCAAAACCCTGTTTCTTCACAGGTTTAGCTTCCTAAGTTACAGCATACATTCATTATCCGACTTTCTGTATTAGACC
TCAGTAAATACCTGAGCCAGTGTTTCCTAATGATTTTCTTTCAATAAACTGATGATTTTCCGGAAGTGACTCTTTATATCTATAAATACTCTCAATATCCTGCATTAATGGTATACCT
GATACAAGCTTCATGCAAAATACCCTCTATAGGAATGCGATCGACTTGTGGGGCATTCATCGGCGATATCCTACTGTAATAGAAAGCACCGGCATAACAATTCCGGTCCCAACTGAAG
TATATCAATGAATTTGATCAGGATTATACTATCAGGAAATCGTTGTTTCCCGAAAAAACAAATACCTAATCCATTTTTCTCAGCCAATGTATCATAGGTCTGAGCCACACGAGCAATGC
ATCTCGTAGTAGCATTCAAAGAAAAGAAAGGATTGTCTAACCCTGGACCGCTTACGCTCTCTGCACCATTTACGTCAGAACGGTCAATTAAAAGGCCACAATAATAAGAGTACAACCG
GAGCACCTGTTGTGGTTGACGAGTGTTAAATGGGCATGCCAATCTCTCTTTCAAATCTTGTCACGAGTCACAATCACACTATATAATTGCATATATATACCCATTGTAGAAGCTTCGACG
GAACCCCTTTTCAAGCAAATCATTTAACAAACAAGCAGAAATATCTTCCTGCTTATATTTACGGAGTCCGTCTATTATTTATCGATTCCATATAGTATTTTTCTTGCAAGATAAAATA
AGAAAAGAGTAATATTGCATTTTTTGCAAAATATTATATGCCAAGCTGGCCTTTTCAACTCGTCAATCACTCATCCCCTCCATCTAGATTTTCAAAACAGAAAACTTCTACAAGGATA
AGCAAAGCATAATATATCTTTTGATAATCAATAAGTTAACAAATATGCTTGGATAACTCAATTTGGTTGAATACATTACTCTGCTTTTTATTTTGTTAAATTGGAAAAACATATATAT
CGCTTTGCCGAAACTTATCTATTATAATGAGTTGTAACTTCTACACTAAAACTGTGTTGATTGATATTACACGTTAATATAAAAATAATAAGGTGGAAAGACAAACGTACACCTATGA
CGAAGCTTTCGAAGCATCTTTACAATACTTCAGAGGTGATGAACTTTGCTGCCAGAGTTTGAAAGGTAAACAAATAGTGCGCTAAAAGATTCTTTGGGAACATATAGATAATGTGTCT
CGCCCGAAGATATGCATTGGCGCATTGCAAATGAAGTAGCCCGCATTGAAGCTAAATATAAGAACGGGTTAAACGCACAGCAGCTTTACGAATTATTGGATCACTTTAAATACATTGT
TCCGCAAGGTAGCCTATGACAGGTATCGGGTAACGACTTTCAAGTGCTTCCTTATCAAACTCTTTGTCATTGGAATTGAGGAGCAGCCGCACGTCTGTAAACGGAGGTATAATCCGCA
```

Given reads, we want to find which ones “overlap”

ACGTAGATAGCATGCTAGCAGCATGCTAGCA

GCATGCTAGCACGTAGATAGCATGCTAGCA

ATGCTAGCAGCATGCTAGCACGTAGATAGCATGCTAGCA

TGGATAAGATAGCATGCTAGCGATAGATCAAATGCTAGCAG

GCATGCTAGCAAGTACATGGATAAGATAGCATGCTAGCGATAG

Given reads, we want to find which ones "overlap"

ACGTAGATAGCATGCTAGCAGCATGCTAGCA

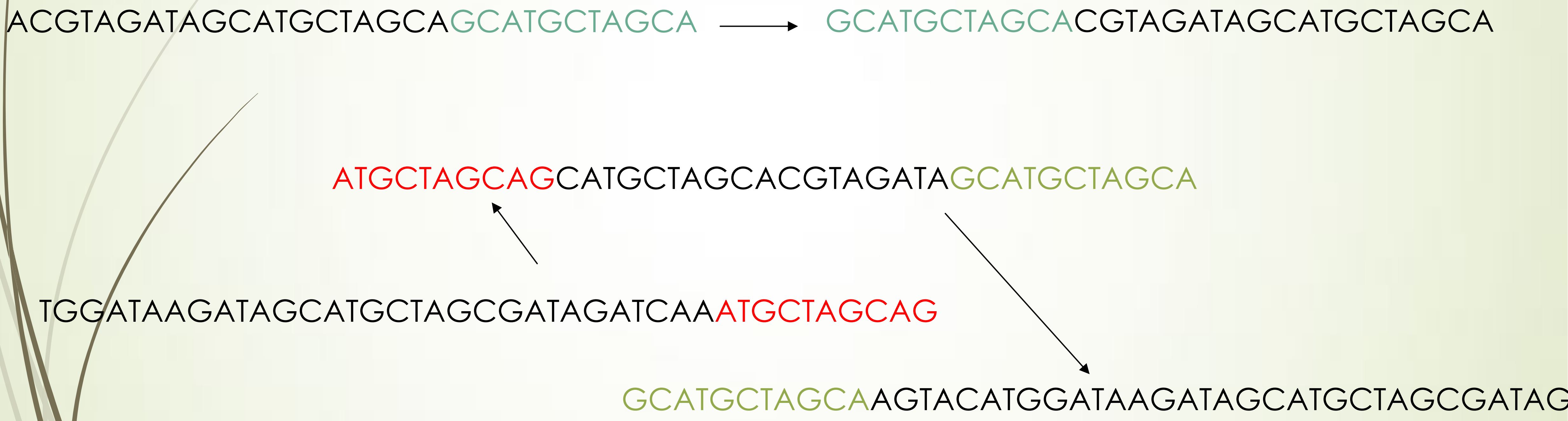
GCATGCTAGCACGTAGATAGCATGCTAGCA

ATGCTAGCAGCATGCTAGCACGTAGATAGCATGCTAGCA

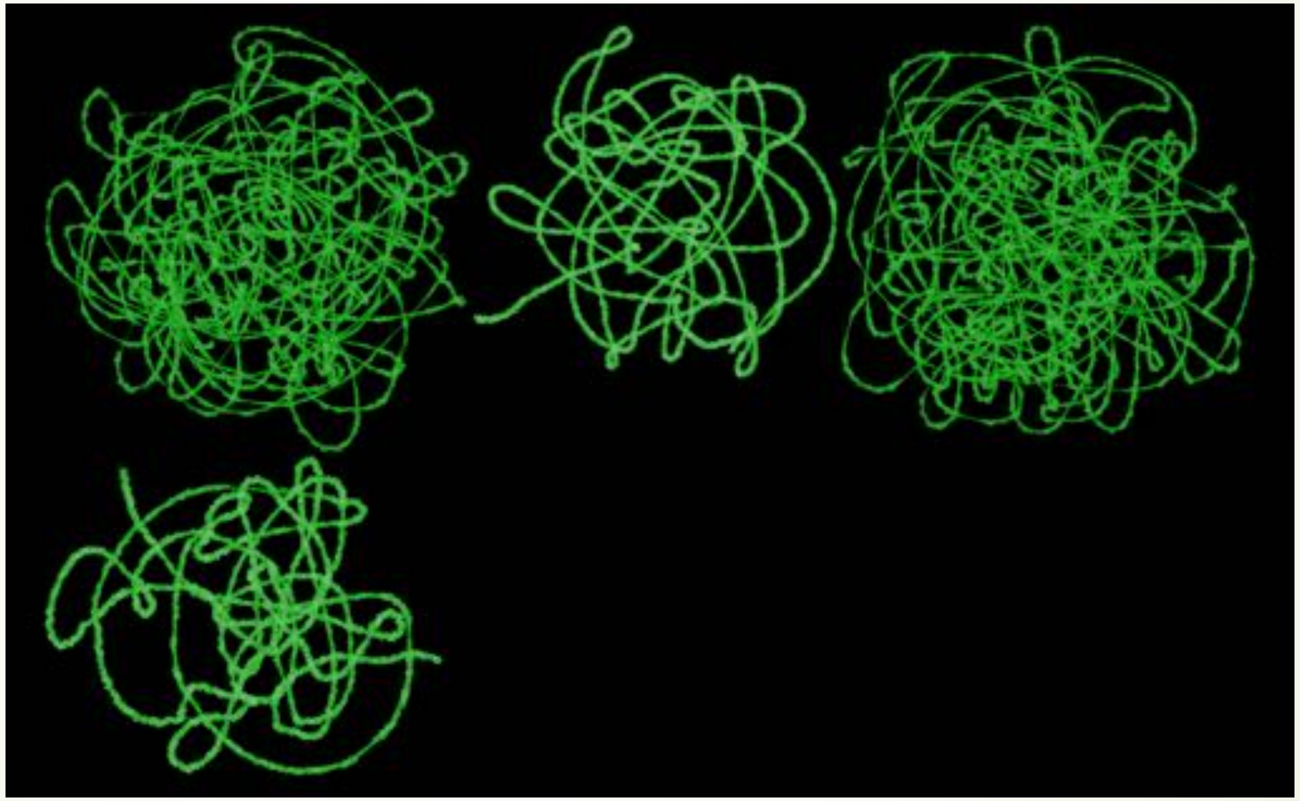
TGGATAAGATAGCATGCTAGCGATAGATCAAATGCTAGCAG

GCATGCTAGCAAGTACATGGATAAGATAGCATGCTAGCGATAG

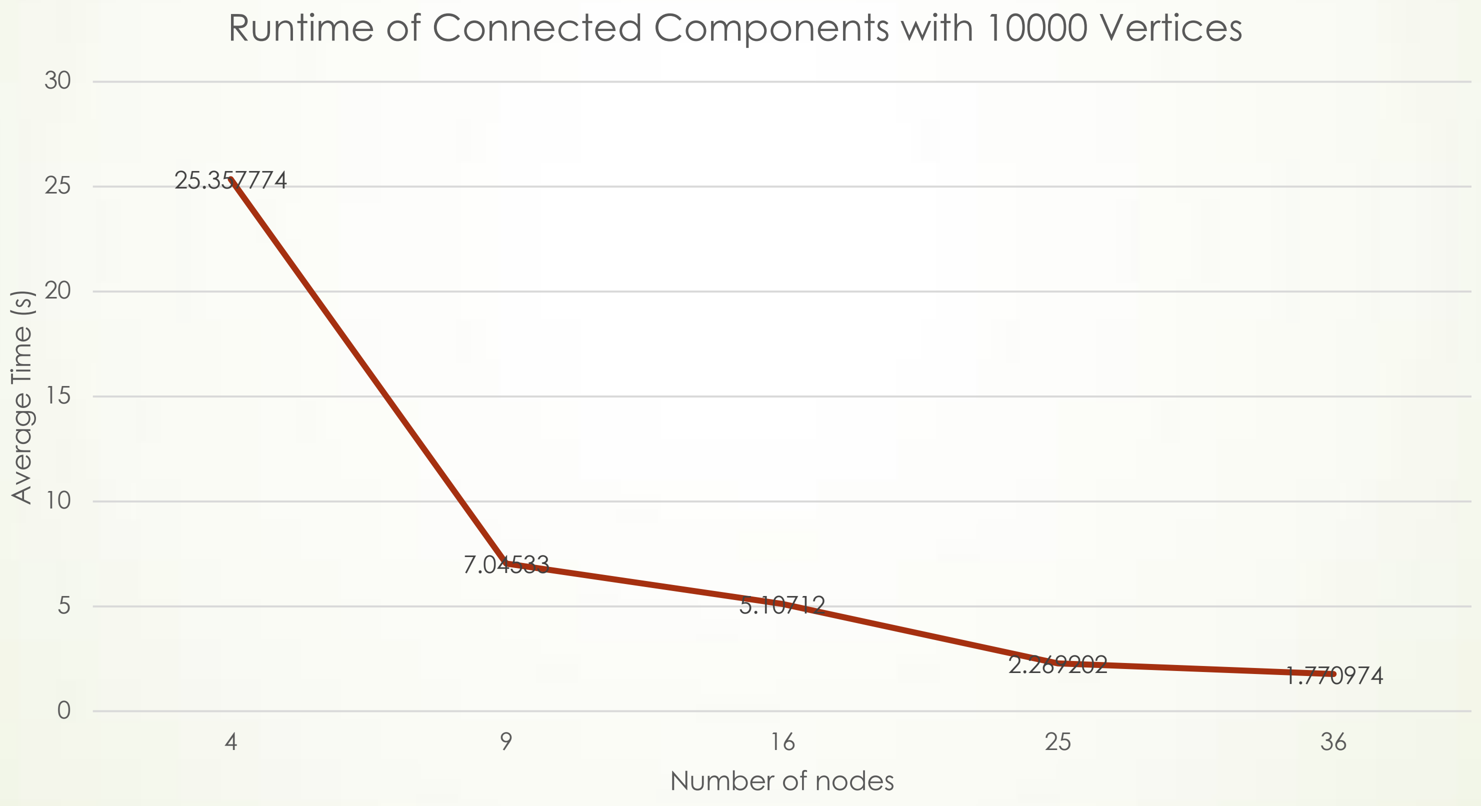
Given reads, we want to find which ones “overlap”


ACGTAGATAGCATGCTAGCAGCATGCTAGCA
GCATGCTAGCACGTAGATAGCATGCTAGCA
ATGCTAGCAGCATGCTAGCACGTAGATAGCATGCTAGCA
TGGATAAGATAGCATGCTAGCGATAGATCAAATGCTAGCAG
GCATGCTAGCAAGTACATGGATAAGATAGCATGCTAGCGATAG

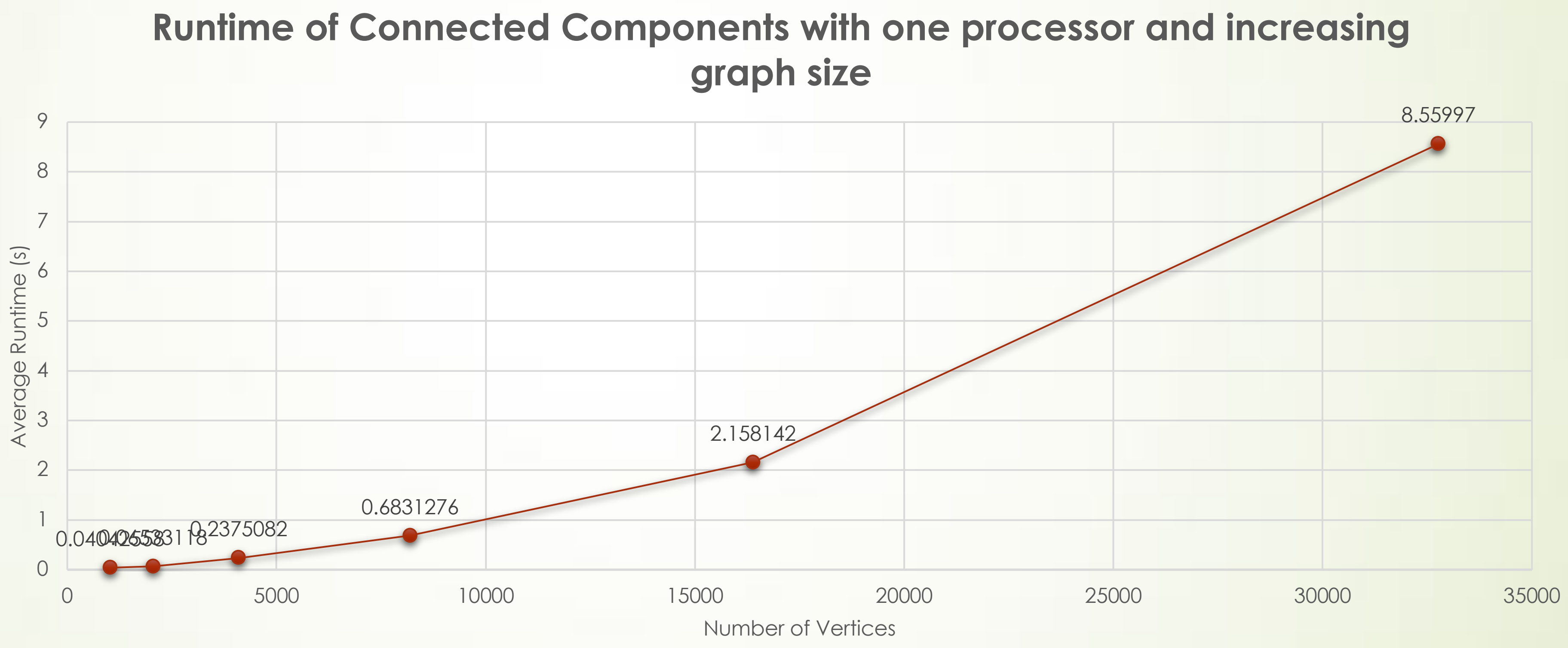
Visualization of Overlap Graph



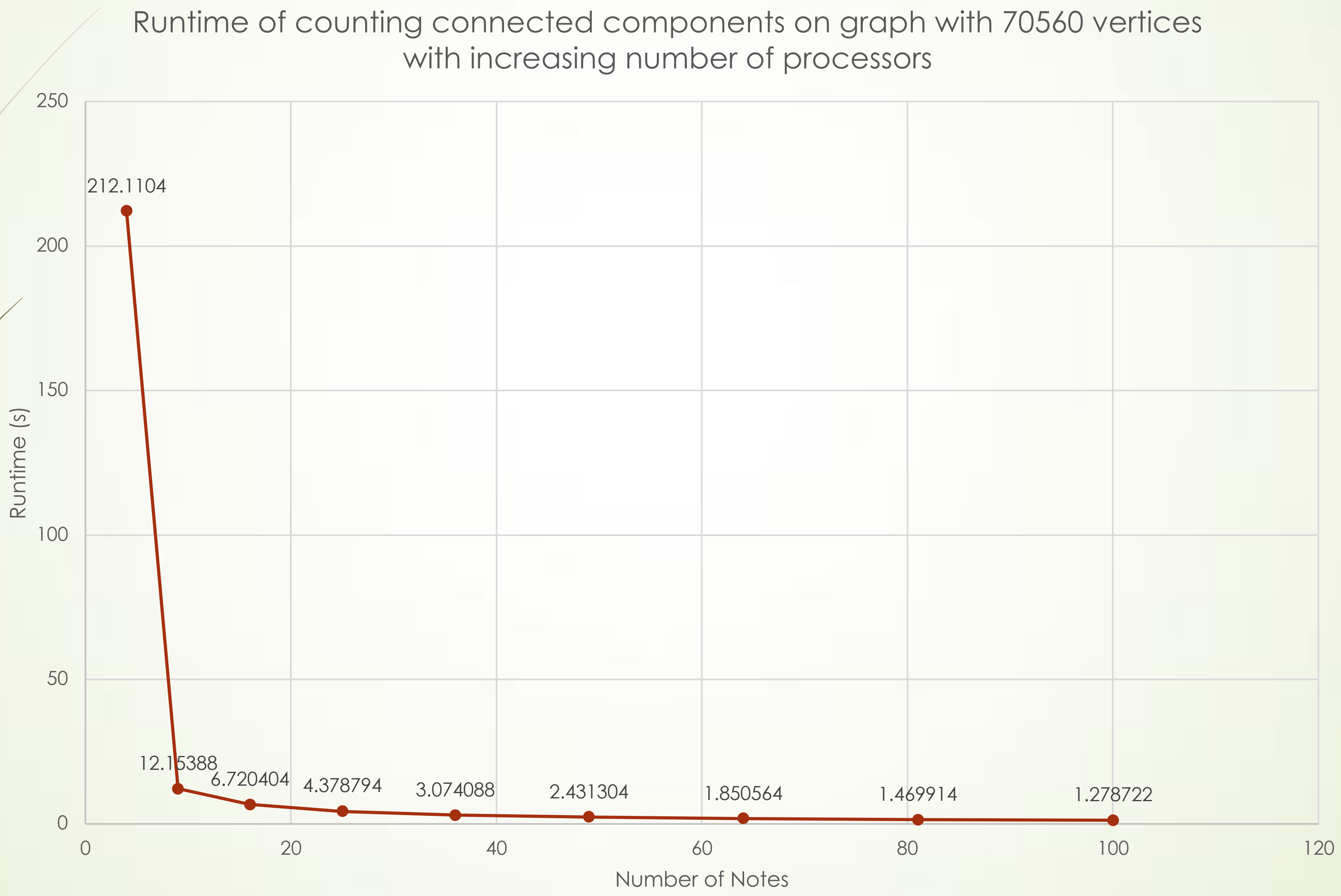
Runtime on overlap graph which has 10000 nodes



Runtime on single processor with increasing data size



Constant data size on multiple processors



Learning outcomes

- Different servers can give you dramatically different runtimes, so try to run all experiments on the same server
- Graph structure can also affect runtime due to different convergence times [3].
- Always use a seed when running experiments on random models
- Biological data can be a pain to work with

References

1. Kumar, S., S. Goddard, and J. Prins. *Connected components algorithms for mesh-connected parallel computers*. AMS, 1997.
2. Flick, Patrick, et al. "A parallel connectivity algorithm for de Bruijn graphs in metagenomic applications." Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2015.
3. Howe, Adina Chuang, et al. "Tackling soil diversity with the assembly of large, complex metagenomes." *Proceedings of the National Academy of Sciences* 111.13 (2014): 4904-4909.
4. JáJá, Joseph (1992). *An Introduction to Parallel Algorithms*. Addison Wesley.
5. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001) [1990]. Introduction to Algorithms (2nd ed.). MIT Press and McGraw-Hill.https://en.wikipedia.org/wiki/Pointer_jumping

Questions?