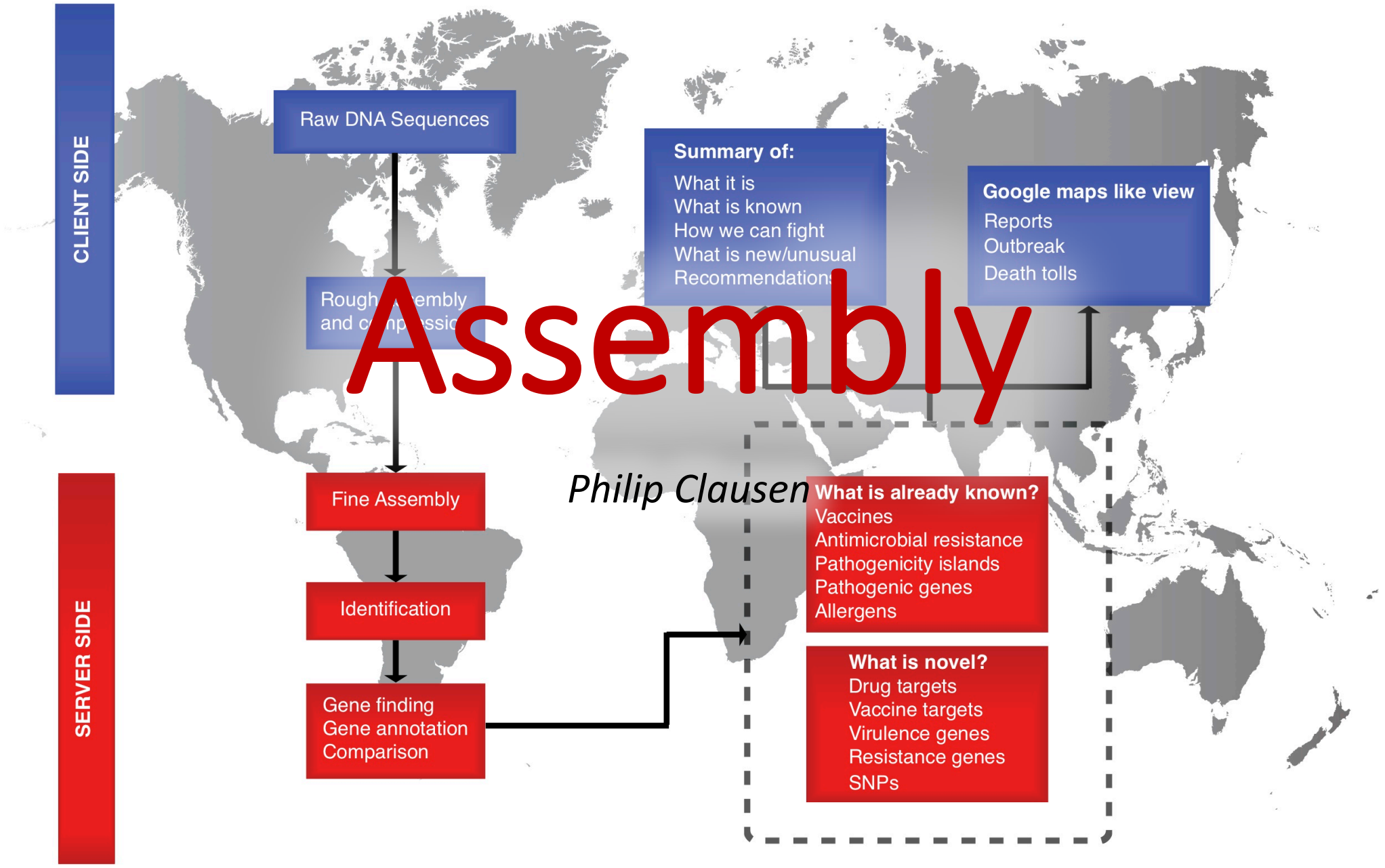


Assembly

Philip Clausen

Agenda:

1. **What**
2. **Why**
3. **How**

1. What

```
@ILLUMINA-3BDE4F_0027:2:1:10636:1724#GCCAAT/1
TGCACAGGTAGCCCCTACGCCGCGNATGAACGACCGGAAACGCCGTCACA
+
a^DG_DPGGDQDFFFFQFKD\\Y\Bab]ac[NY[acaXa]cc_ccYcccc
@ILLUMINA-3BDE4F_0027:2:1:8882:2205#GCCAAT/1
TCCGAATTAAGCGCATATTCTGGTNCACGACTTTGAAGTGTCGCCCTTTT
+
fffffdfffdfcfffdacef_ffcccfcef`WWW^^W^addeccYccbb^
@ILLUMINA-3BDE4F_0027:2:1:14729:12149#GCCAAT/1
TTCCGGCGCATGAGTGTGTATCTTNGCTTTCAACATATTCCCGGGCAATG
+
^^EF[FKFKFZFFFFFQJQF_]__BcaaaccI[`^ca\abccccccccccc
@ILLUMINA-3BDE4F_0027:2:2:15011:1176#GCCAAT/1
TGCGGGATATTATTAAACACATCAAGGCACAGCGCCTTATCTAAACAACTA
+
gggggfbfaGSZK]NKZSRQcccddgfgfgggggggcggggffdgggegg
```

```
>NODE_1_length_842323_cov_19.7952_ID_4668
TCGGATCAAGGCAGTCACTATACCAGCAGGAATTTCAGACAGTTACTGTGG
AGATATCAGATAAAGCAAAGCCTGAGTCGCCGGGGAAATTGCTGGGATAAC
AGCCCAATGGAACGGTTCTTCAGAAGCCTGAAAACAGAGTGGGTACCGGAT
AATGGCTACGCGAATTTTAGCGAAGCCAGCACGGCAATAACGAATTACATC
ACAGGATATTACAGCCAGCTCAGACCTCATCAATATAATGGTGGTTTGACG
CCGAATGAATCAGAACGATTGTTCTGGAAAAACTCTAAAGCTGTGGCCAGT
TTTTGTTGACCACTACACTCCCGGATTGATGTAGTCAGTAAAAGTATGGCA
CTGACTACTTTAGGAAGGGCGTCCATCACATCACTACAATTTTTATGAACA
ATTTTTGTTGTGAATATTACTCTTGCAATATGATGTTGATCTATCCAGATA
TAAGATCTTATCTGCAATTCGTAGTGCCTCACCTTTATGGCTTATAATGAT
TTTTGTACAGGGAATAGTGGAAATATAATCAATAACTTGTGTTTCAGATTC
ATGGTCAAGATTACTCGTTGCTTCATCAAGGATAAGCAAAGCTGGTTTTCT
GTATATAGCTCTGGCAAGTAAAATACGCTGAATCTGACCAACAGATAAAAA
AGCATTAATATCTGTCACTTGGGAGTAATATCCCATAGGTAGACTATTGAT
TACGTCATGTATCCCCAGACCTTTTGTAATATGAATAACATCATCAAGAGA
AACAGTATTATCAAAGGAAATATTATAAAGCAGA
```

2. Why

X GB

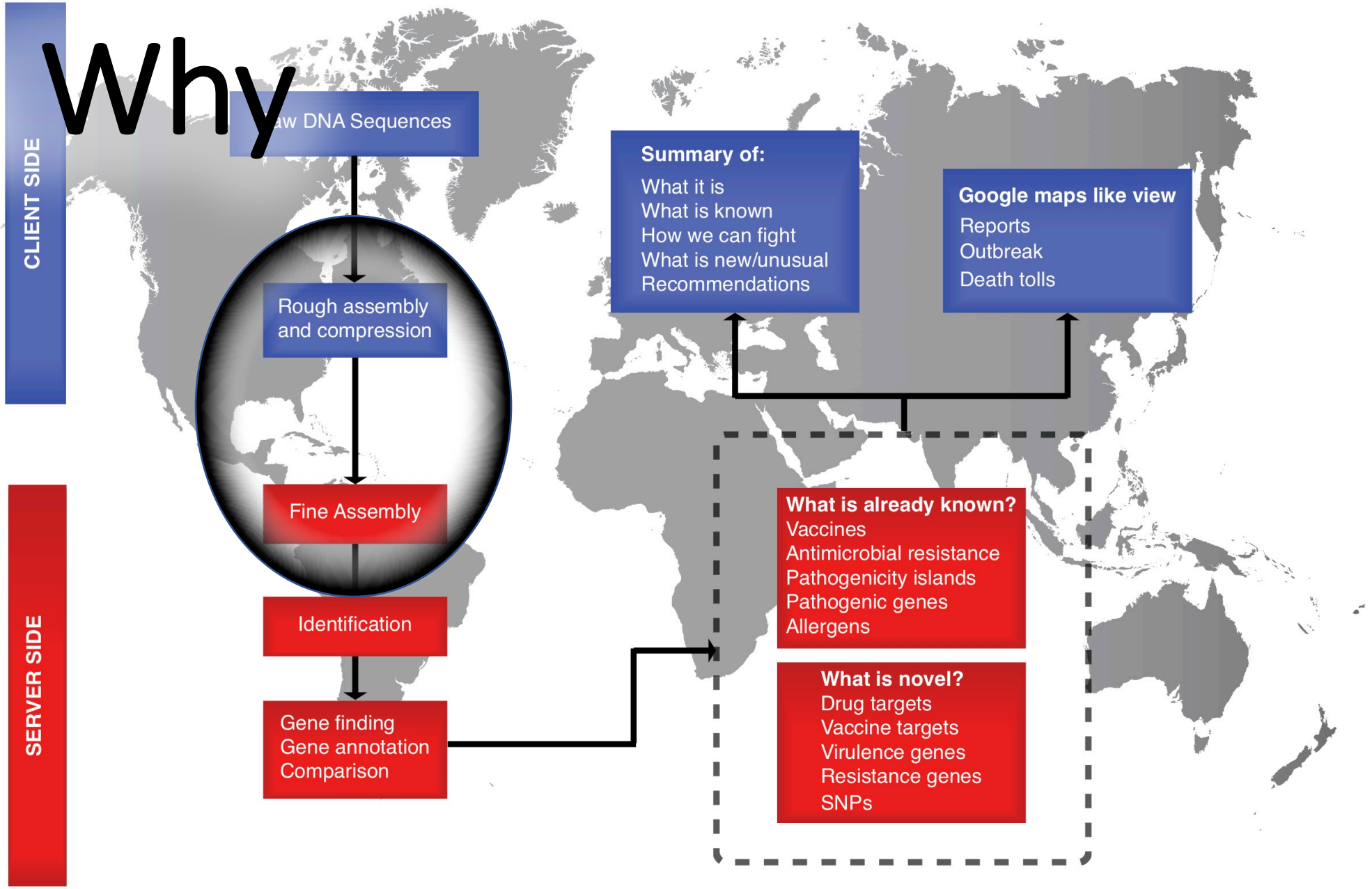
```
@ILLUMINA-3BDE4F_0027:2:1:10636:1724#GCCAAT/1
TGCACAGGTAGCCCCTACGCCGCGNATGAACGACCGGAAACGCCGTCACA
+
a^DG_DPGGDQDFFFFQFKD\\Y\Bab]ac[NY[acaXa]cc_ccYcccc
@ILLUMINA-3BDE4F_0027:2:1:8882:2205#GCCAAT/1
TCCGAATTAAGCGCATATTCTGGTNCACGACTTTGAAGTGTCGCCCTTTT
+
fffffdfffdfcfffdacef_ffcccfcef`WWW^^W^addeccYccbb^
@ILLUMINA-3BDE4F_0027:2:1:14729:12149#GCCAAT/1
TTCCGGCGCATGAGTGTGTATCTTNGCTTTCAACATATTCCCGGGCAATG
+
^^EF[FKFKFZFFFFFQJQF_]__BcaaaccI[`^ca\abccccccccccc
@ILLUMINA-3BDE4F_0027:2:2:15011:1176#GCCAAT/1
TGCGGGATATTATTAAACACATCAAGGCACAGCGCCTTATCTAAACAACTA
+
gggggfbfaGSZK]NKZSRQcccddgfgfgggggggcggggffdgggegg
```

Y=19.8

X / Y GB

```
>NODE_1_length_842323_cov_19.7952_ID_4668
TCGGATCAAGGCAGTCACTATACCAGCAGGAATTTCAGACAGTTACTGTGG
AGATATCAGATAAAGCAAAGCCTGAGTCGCCGGGGAAATTGCTGGGATAAC
AGCCCAATGGAACGGTTCTTCAGAAGCCTGAAAACAGAGTGGGTACCGGAT
AATGGCTACGCGAATTTTAGCGAAGCCAGCACGGCAATAACGAATTACATC
ACAGGATATTACAGCCAGCTCAGACCTCATCAATATAATGGTGGTTTGACG
CCGAATGAATCAGAACGATTGTTCTGGAAAAACTCTAAAGCTGTGGCCAGT
TTTTGTTGTCACTACACTCCCGGATTGCTGAGTCGGTAAAGTATGGCA
CTGACTACTTAGGAAGGGCGTCCATCACAAACTAAATTTTATGAACA
ATTTTTGTTTTGAATATTACTCTTCAATAAGATTTCATCCAGATA
TAAGATCTTATCTGCAATTCGTAGTGCCTCACCTTTATGGCTTATAATGAT
TTTTGTACAGGGAATAGTGGAAATATAATCAATAACTTGTGTTTCAGATTC
ATGGTCAAGATTACTCGTTGCTTCATCAAGGATAAGCAAAGCTGGTTTTCT
GTATATAGCTCTGGCAAGTAAAATACGCTGAATCTGACCAACAGATAAAAA
AGCATTAATATCTGTCACTTGGGAGTAATATCCCATAGGTAGACTATTGAT
TACGTCATGTATCCCCAGACCTTTTGTAATATGAATAACATCATCAAGAGA
AACAGTATTATCAAAGGAAATATTATAAAGCAGA
```

2. Why


CLIENT SIDE
SERVER SIDE
aw DNA Sequences
Rough assembly and compression
Fine Assembly
Identification
Gene finding
Gene annotation
Comparison
Summary of:
What it is
What is known
How we can fight
What is new/unusual
Recommendations
Google maps like view
Reports
Outbreak
Death tolls
What is already known?
Vaccines
Antimicrobial resistance
Pathogenicity islands
Pathogenic genes
Allergens
What is novel?
Drug targets
Vaccine targets
Virulence genes
Resistance genes
SNPs

3. How

1. Assemble reads into contigs, by identifying overlaps between reads.

2. Order and join contigs in order to form longer scaffolds.

de Bruijn graph

1. Each read is broke down to *k*-mers, which can be searched for in constant time O(*k*).

2. The *k*-mers are connected to their leading and trailing *k*-mer, to form a network of *k*-mers.

3. The optimal path through the network is found (Eulerian if possible), and the *k*-mers along the path is connected in order to form contigs.

Example with $k = 3$;

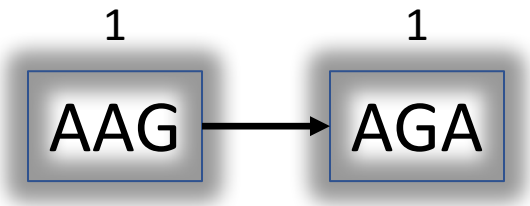
AAGACTCCGACTGGGACTTT

1

AAG

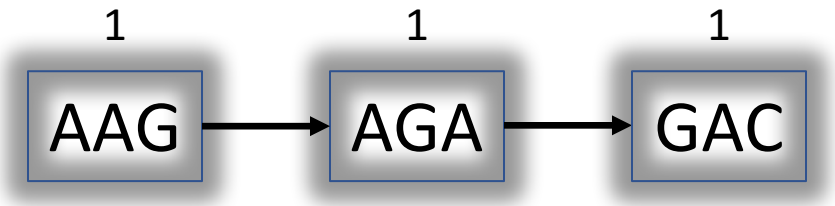
Example with $k = 3$;

AAGACTCCGACTGGGACTTT



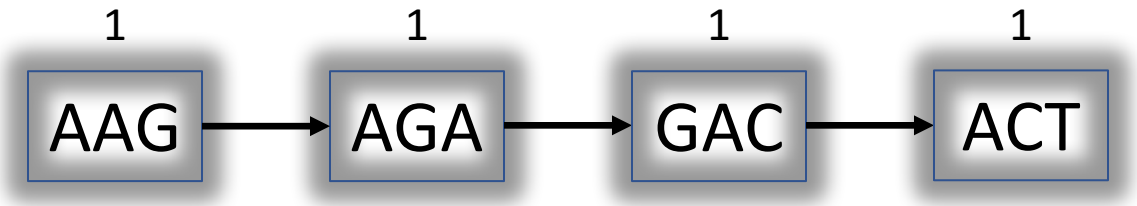
Example with $k = 3$;

AAGACTCCGACTGGGACTTT



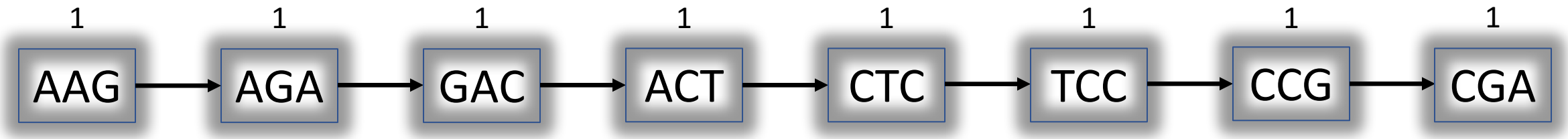
Example with *k* = 3;

AAGACTCCGACTGGGACTTT



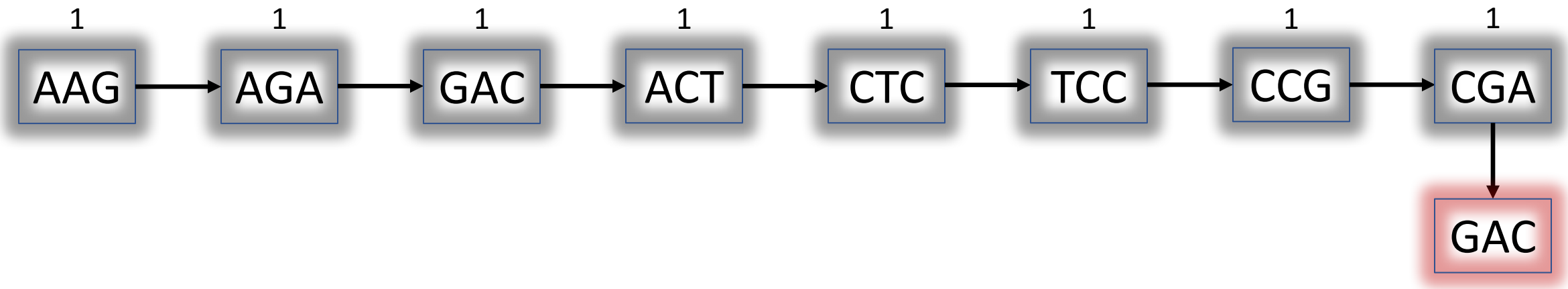
Example with *k* = 3;

AAGACTCCGACTGGGACTTT



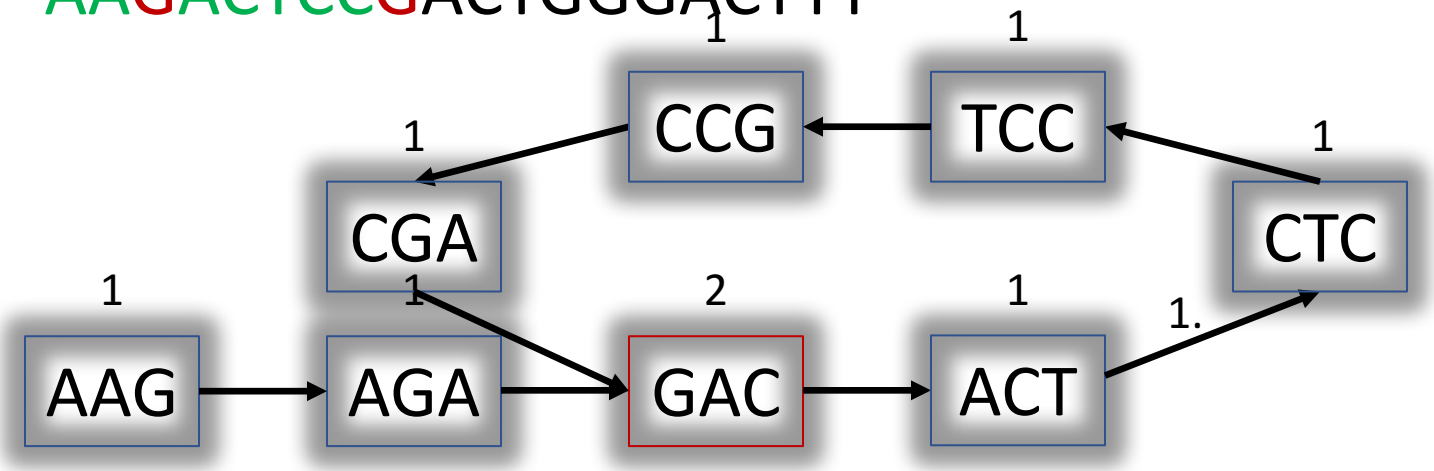
Example with *k* = 3;

AAGACTCCGACTGGGACTTT



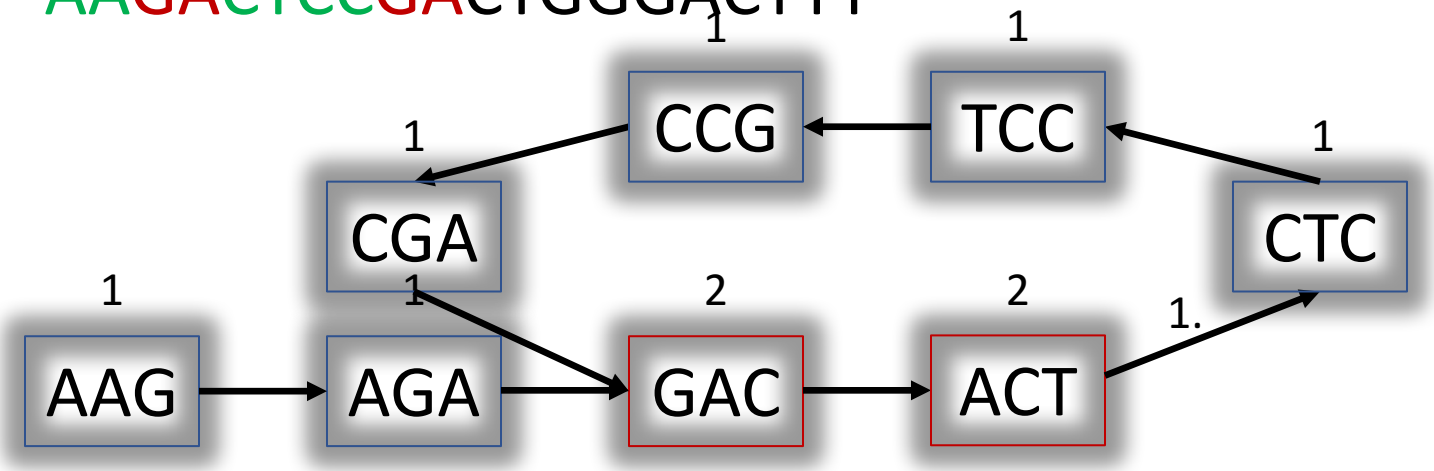
Example with *k* = 3;

AAGACTCCGACTGGGACTTT



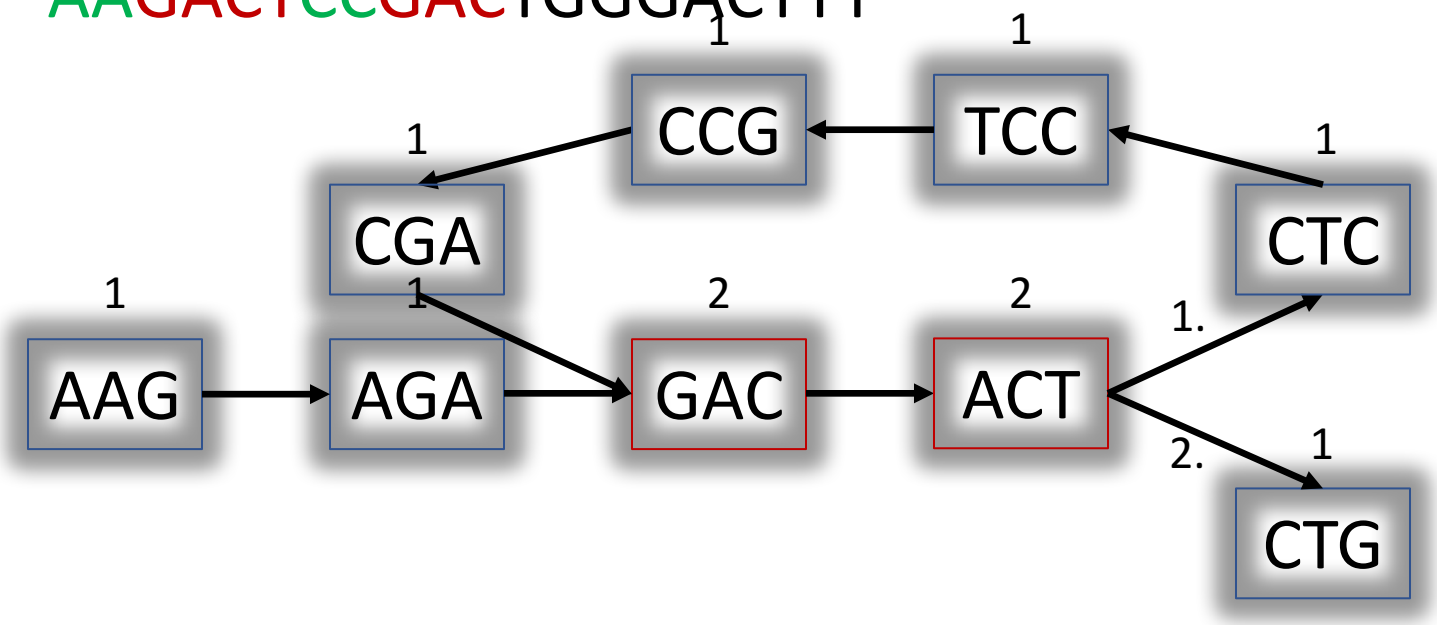
Example with *k* = 3;

AAGACTCCGACTGGGACTTT



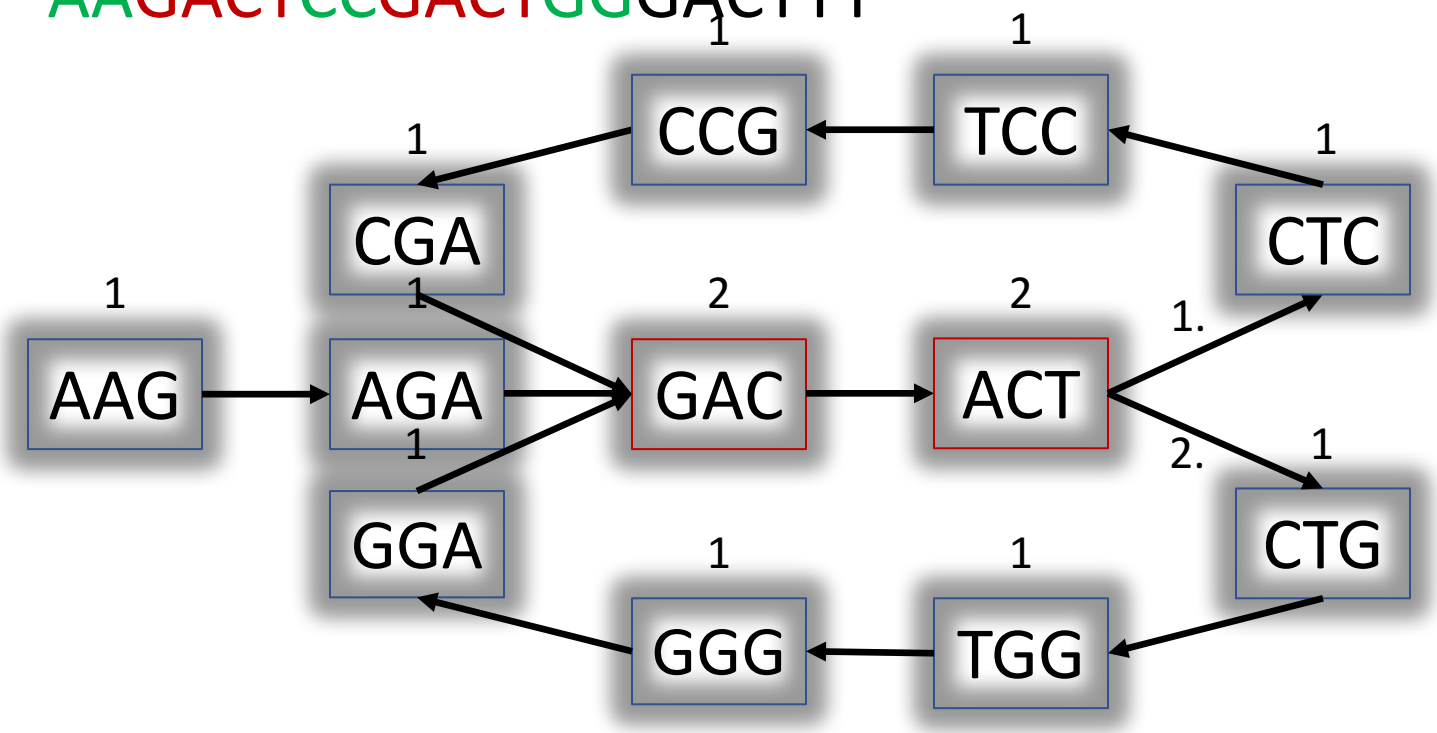
Example with *k* = 3;

AAGACTCCGACTGGGACTTT



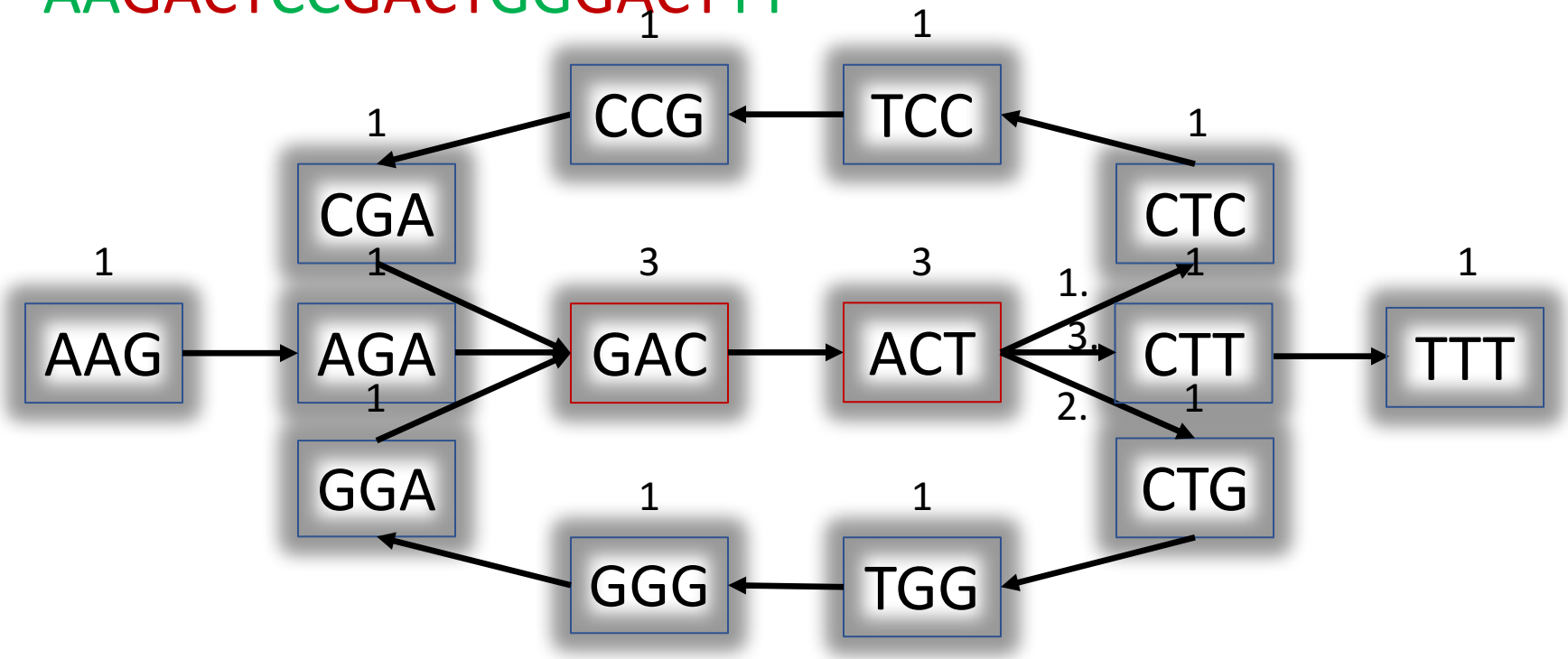
Example with *k* = 3;

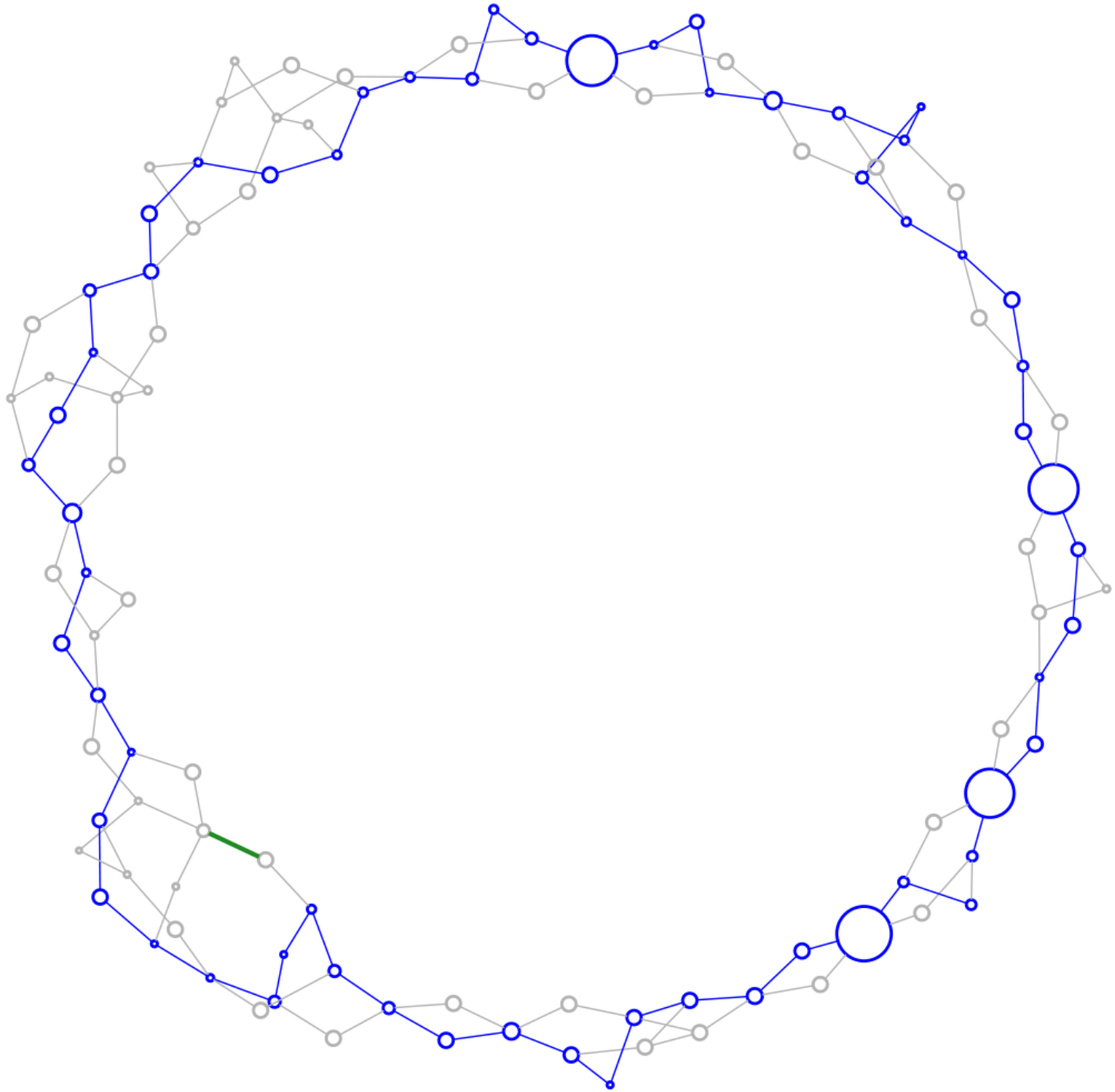
AAGACTCCGACTGGGACTTT

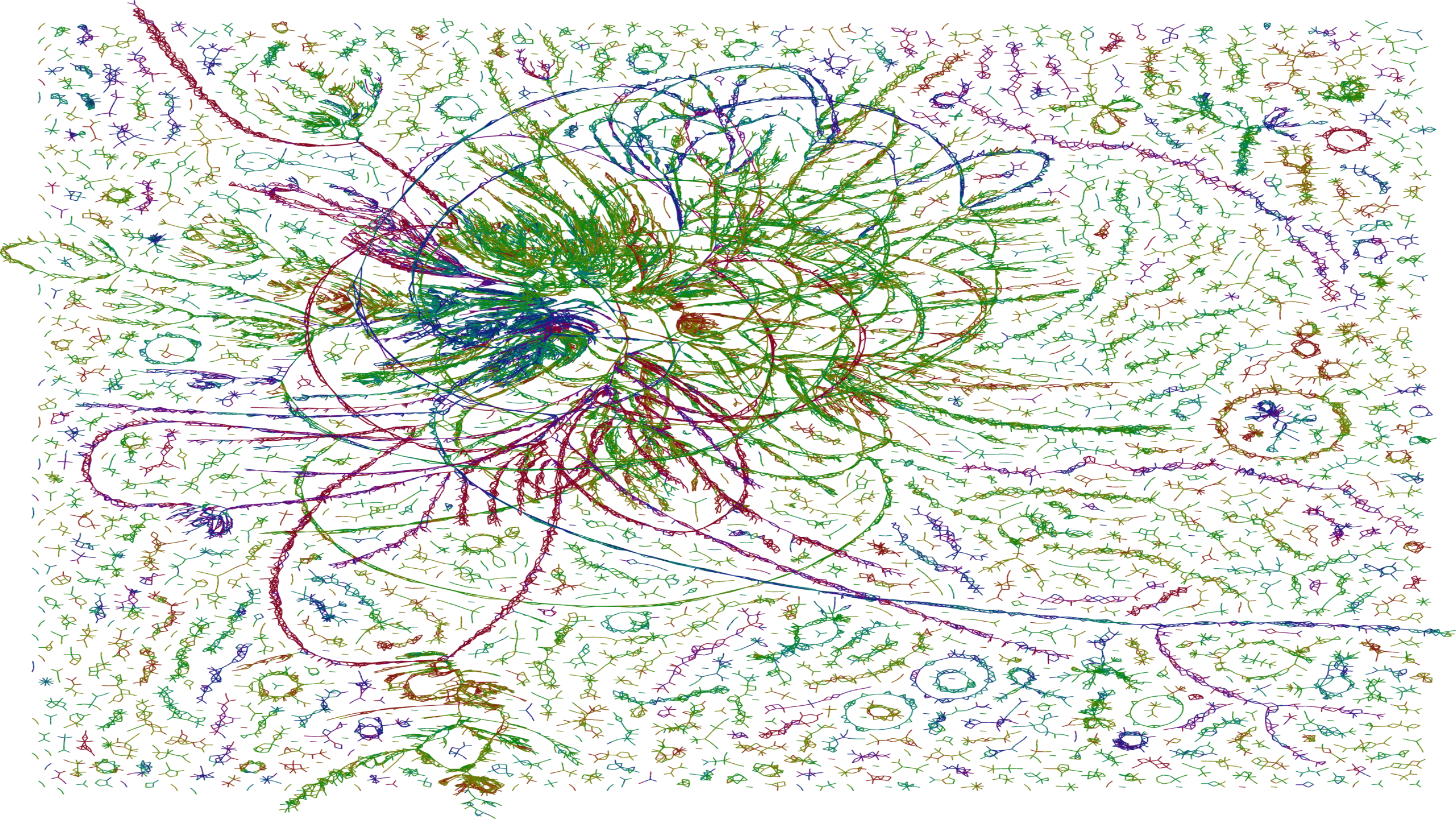


Example with *k* = 3;

AAGACTCCGACTGGGACTTT

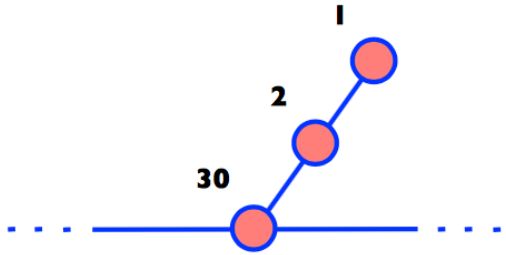




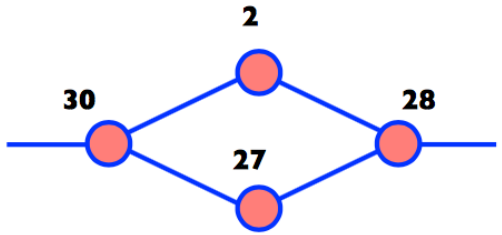


Basic de Bruijn trimming.

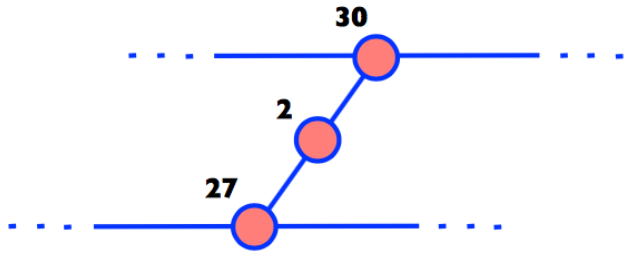
Clip tips
(seq err, end)



Pinch bubbles
(seq err, middle, SNP)



Remove low cov. links



Which k ?

Dependencies of *k*.

1. The depth of your sample.
2. The complexity of the de Bruijn graph.
    1. Contamination.
    2. Repeats.
    3. Plasmids.
    4. Metagenomics.

How do we find *k ?*

Application specific.

1. Spades
2. Velvet
3. Others…

1. Spades

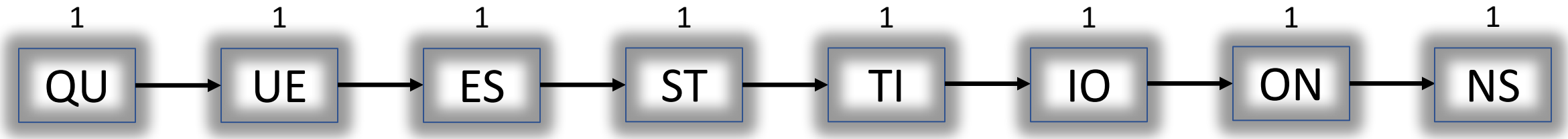
1. Use several sizes of *k*.
2. Allows for assembly at different depths, e.g. plasmids and chromosome.
3. Generally gives a better representation of repeats.
4. Has a tendency to assemble sequence carryover as well.

2. Velvet

1. Makes one assembly, using one *k*.
2. Several tools to identify the optimal *k* is available.
3. Assumes only one organism is present.
4. Handles sequence carryover well.

Test the quality of the assembly.

1. If MLST typing is possible, the assembly is usually of high quality.
2. Look at N50.
3. BLAST a few reads against the assembly.


1
1
1
1
1
1
1
1
QU
UE
ES
ST
TI
IO
ON
NS