Algorithmen der Sequenzanalyse

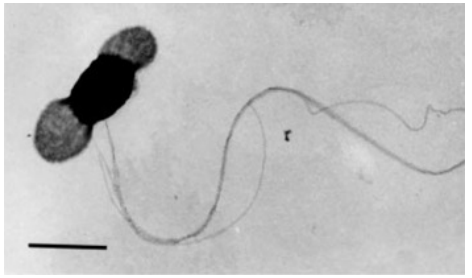
AlgSeq – 21/10/2024

Prof. M. Sammeth

Nachrichten im *oriC* von verschiedenen Bakterien

Im *oriC* von *Thermotoga petrophila*:

aactctatacctccttttgtcgaatttgtgtgatttatagagaaaatcttattaactga
aactaaaatggtaggtttGGTGGTAGGttttgtgtacattttgtagtatctgatttttaa
ttacataccgtatattgtattaaattgacgaacaattgcatggaattgaatatatgcaaa
acaaaCCTACCACCaaactctgtattgaccattttaggacaacttcagGGTGGTAGGttt
ctgaagctctcatcaatagactattttagtctttacaaacaatattaccgttcagattca
agattctacaacgctgttttaatgggcgttgcagaaaacttaccacctaaaatccagtat
ccaagccgatttcagagaaacctaccacttacctaccactaCCTACCACCcgggtggta
agttgcagacattattaaaaacctcatcagaagcttgttcaaaaatttcaatactcgaaa
CCTACCACCtgcgtcccctattatttactactactaataatagcagtataattgatctga



Im *oriC* von *Vibrio cholerae*:

atcaatgatcaacgtaagcttctaagcATGATCAAGgtgctcacacagtttatccacaac
ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca
cggaaagATGATCAAGagaggatgatttcttggccatatcgcaatgaatacttgtgactt
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt
acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga
tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat
tgataatgaatttacatgcttccgcgacgatttacctCTTGATCATcgatccgattgaag
atcttcaattgttaattctcttgcctcgactcatagccatgatgagctCTTGATCATgtt
tccttaaccctctatttttacggaagaATGATCAAGctgctgctCTTGATCATcgtttc

➡ DnaA-boxen sind spezies-spezifisch!

Spezifität von DnaA-Boxen

Hypothese:
Kandidaten des Motives für die DnaA-Box sollten spezifisch für die *oriC*-Region sein und nicht verstreut über das ganze Genom vorkommen.

Muster Findung = Pattern Matching (engl.)

Pattern Matching Problem: *Finde alle Vorkommnisse eines Musters in einem Text.*

Input: Zwei Strings, *Muster* und *Text*.

Output: Alle Positionen in *Text*, an denen ein Substring beginnt, der *Muster* reproduziert.

---

z.B. für *Vibrio Cholerae*

PatternMatching(ATGATCCAG, $Genom_{Vibrio\text{-}cholerae}$) ergibt:
116556, 149355, 151913, 152013, 152394, 186189, 194276, 200076, 224527, 307692, 479770, 610980, 653338, 679985, 768828, 878903, 985368.

Mathemat. Visualisierungen in Python

https://matplotlib.org → API docs
installieren mit “pip install ...”
importieren mit “import ... [as ...]” oder “from ... import ... [as ...]”

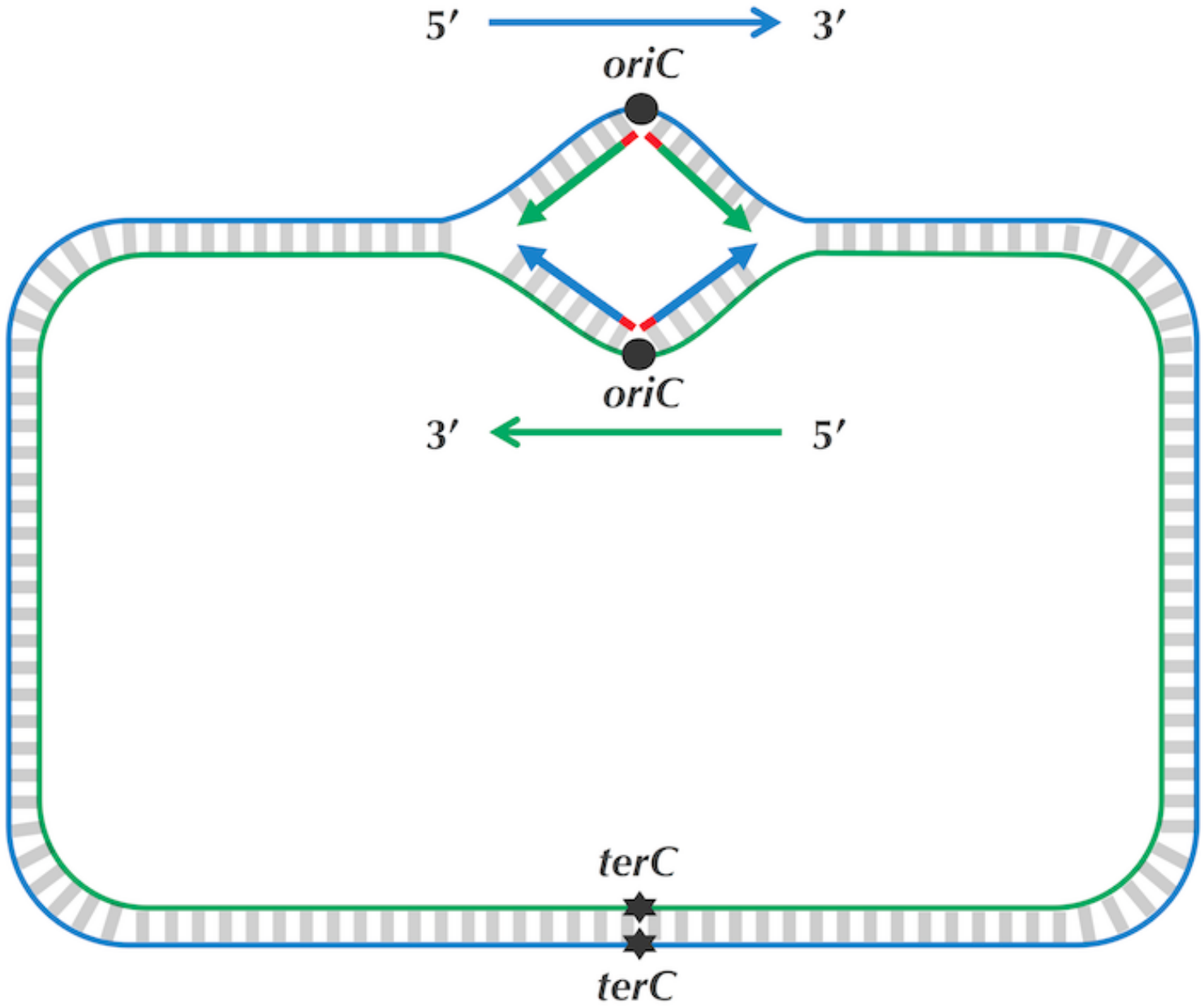
Visualisierung der Spezifität von DnaA-Boxen, z.B.:

- Genom als *horizontale* Linie
- Positionen der kmere als vertikale Linien

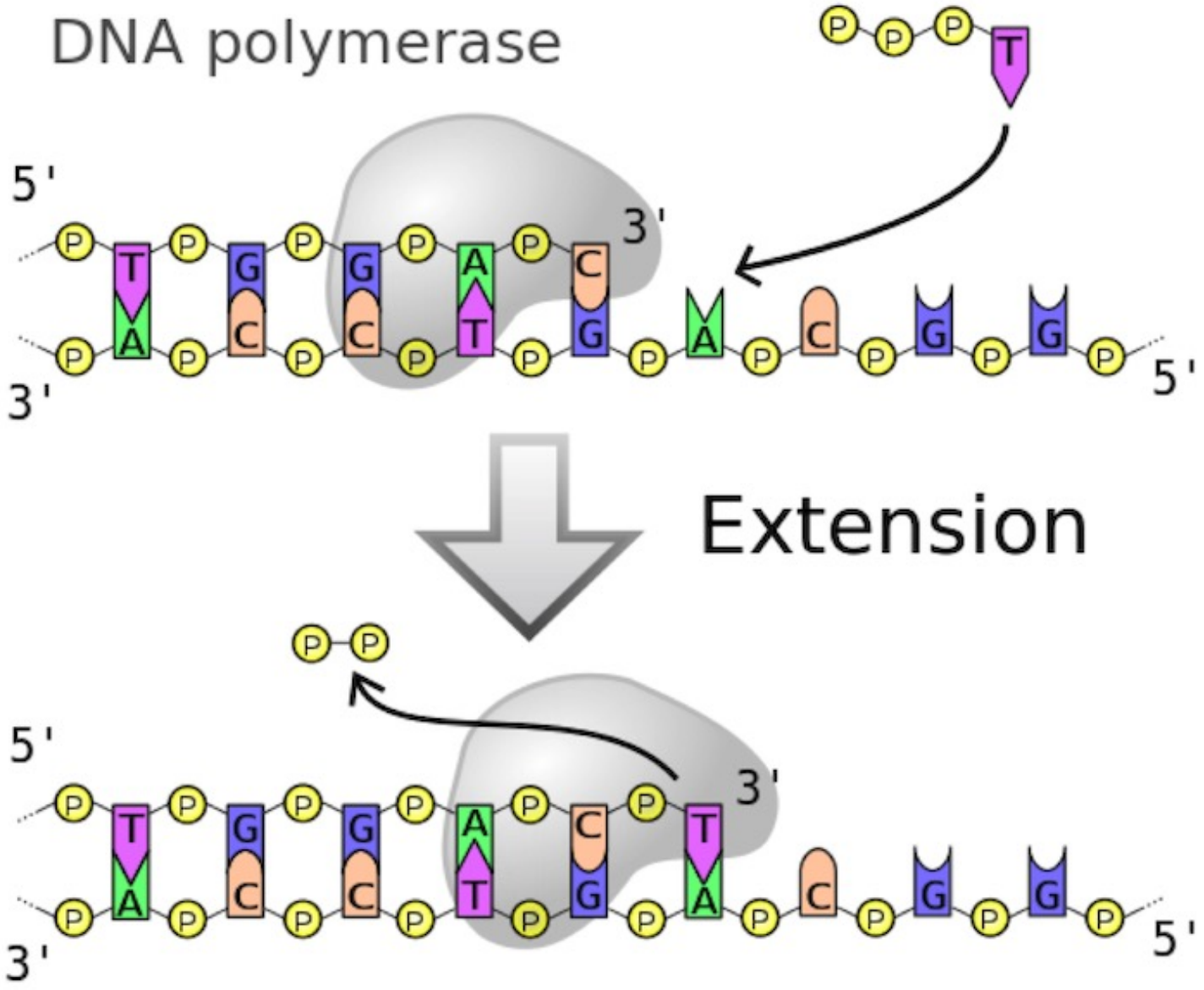
matplotlib.pyplot.vlines()

```
import matplotlib.pyplot as plt
plt.scatter([1,2,3],[1,2,3])
plt.vlines([140000,170000], ymin= -8000, ymax= 5000, colors=["red","green"])
```

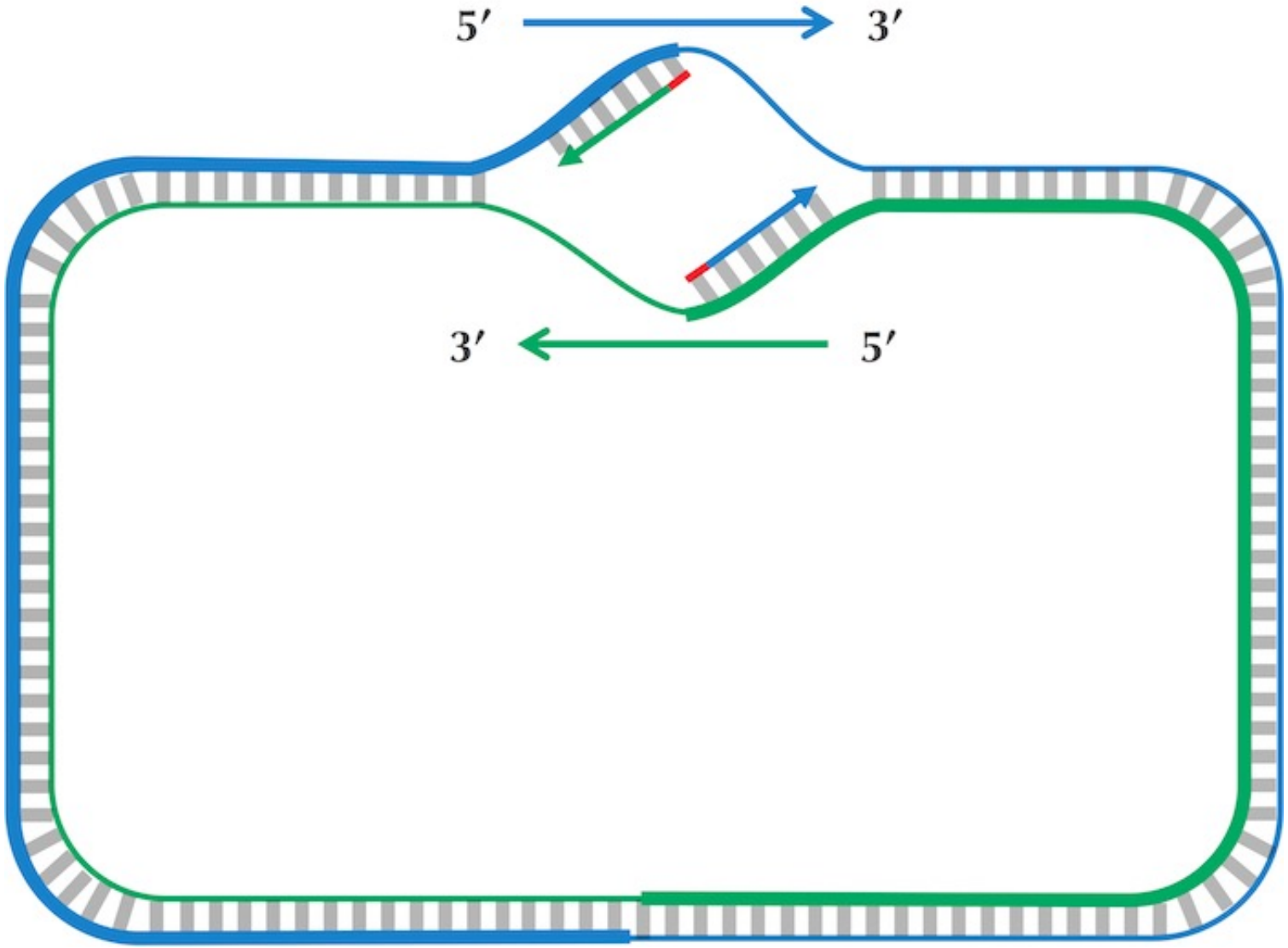
Modell der Replikation in *E. coli*



Nukleotidketten wachsen strikt 5' ➔ 3'

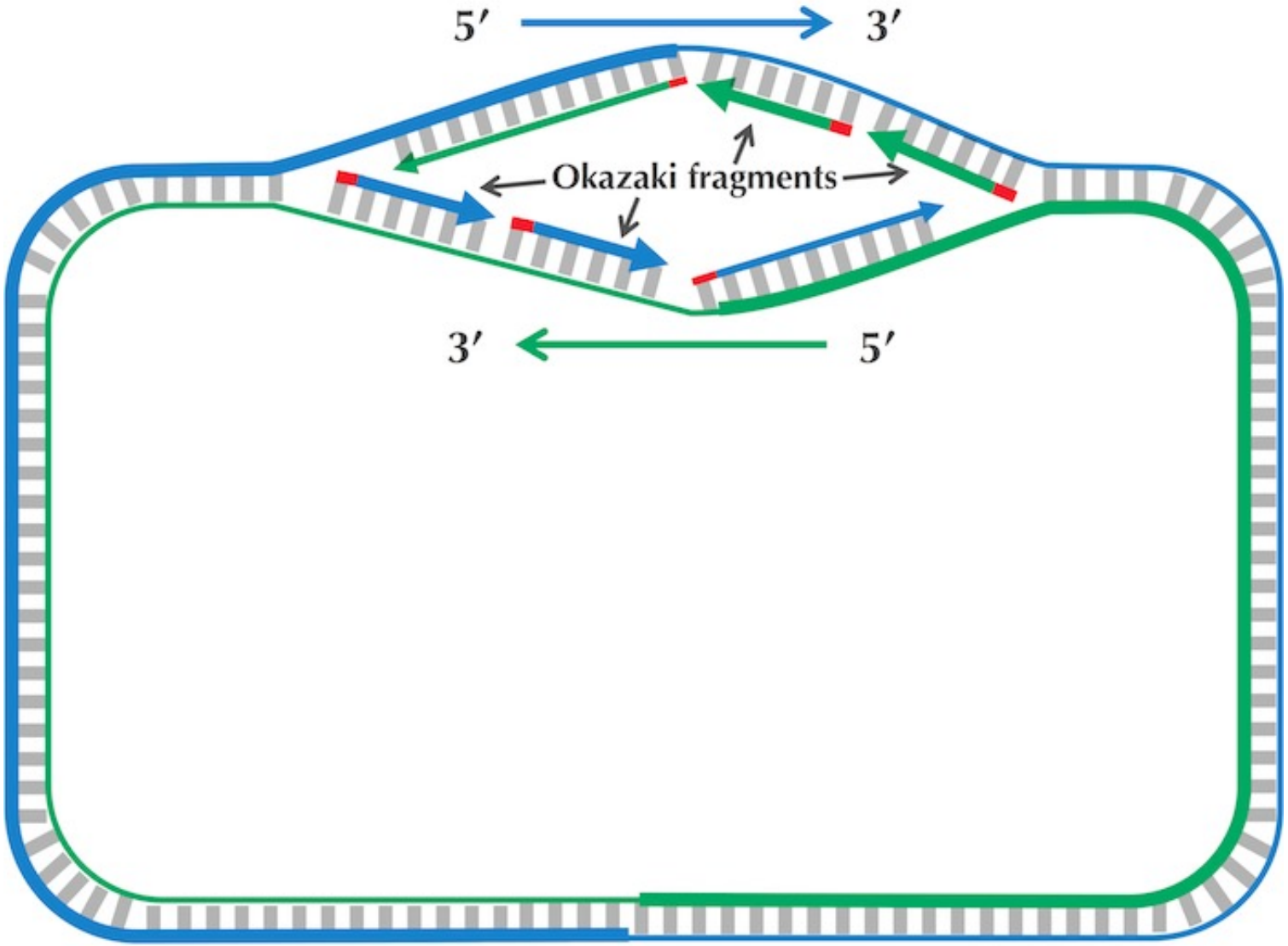


Leitstränge:



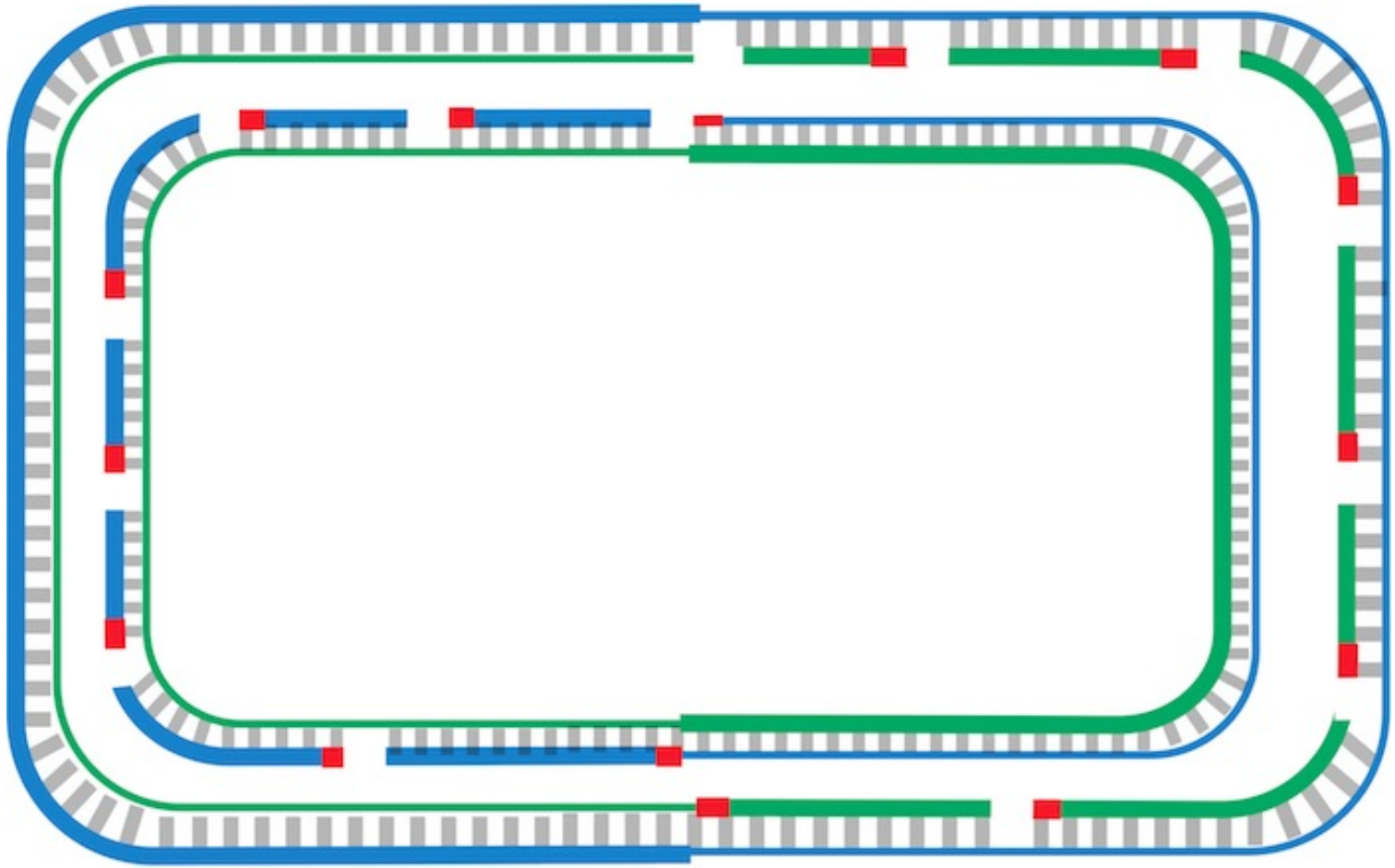
kontinuierliche DNA Synthese 5' ➔ 3'

Folgestränge

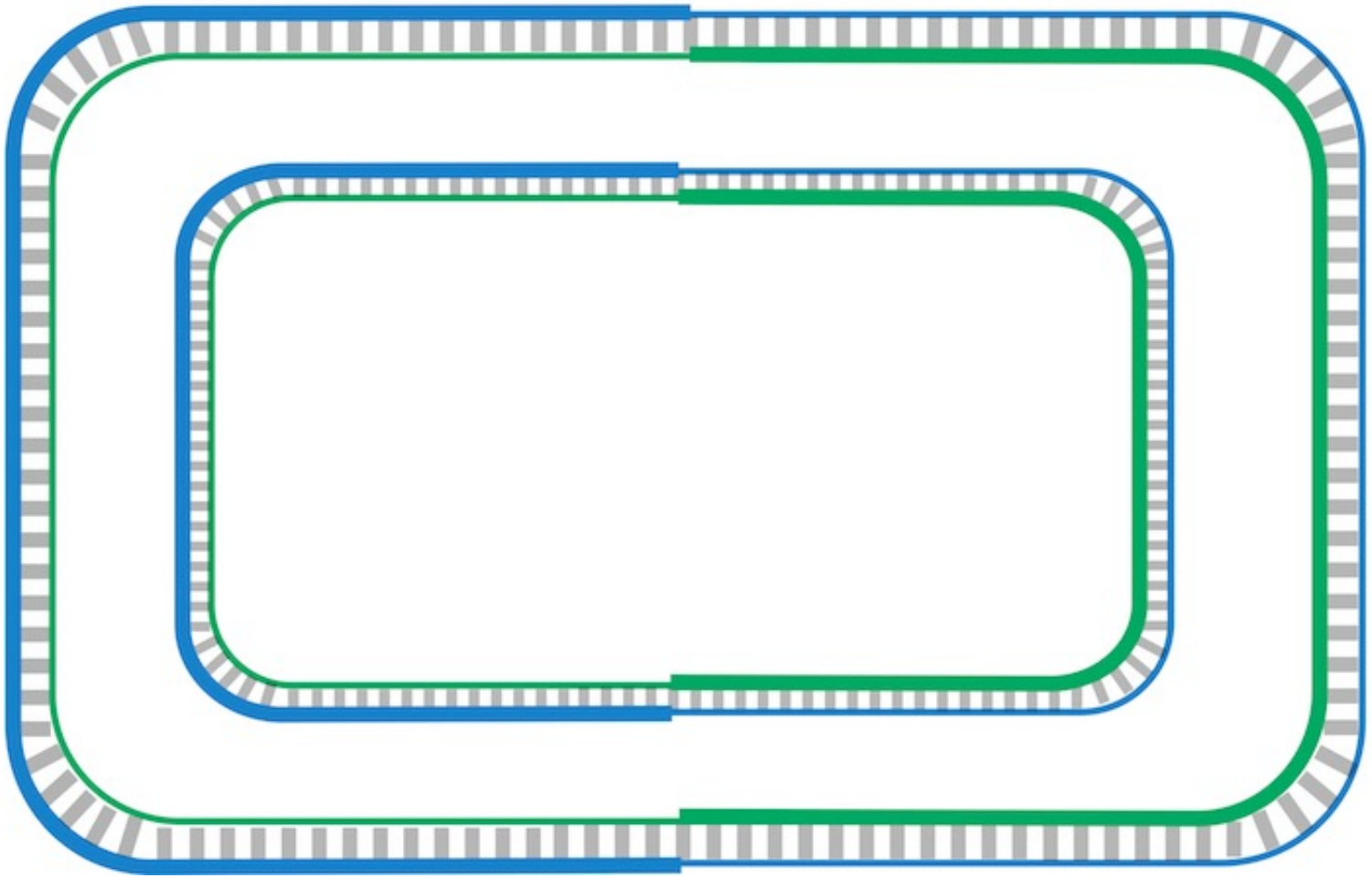


semi-diskontinuierliche DNA Synthese (Okazaki Fragmente)

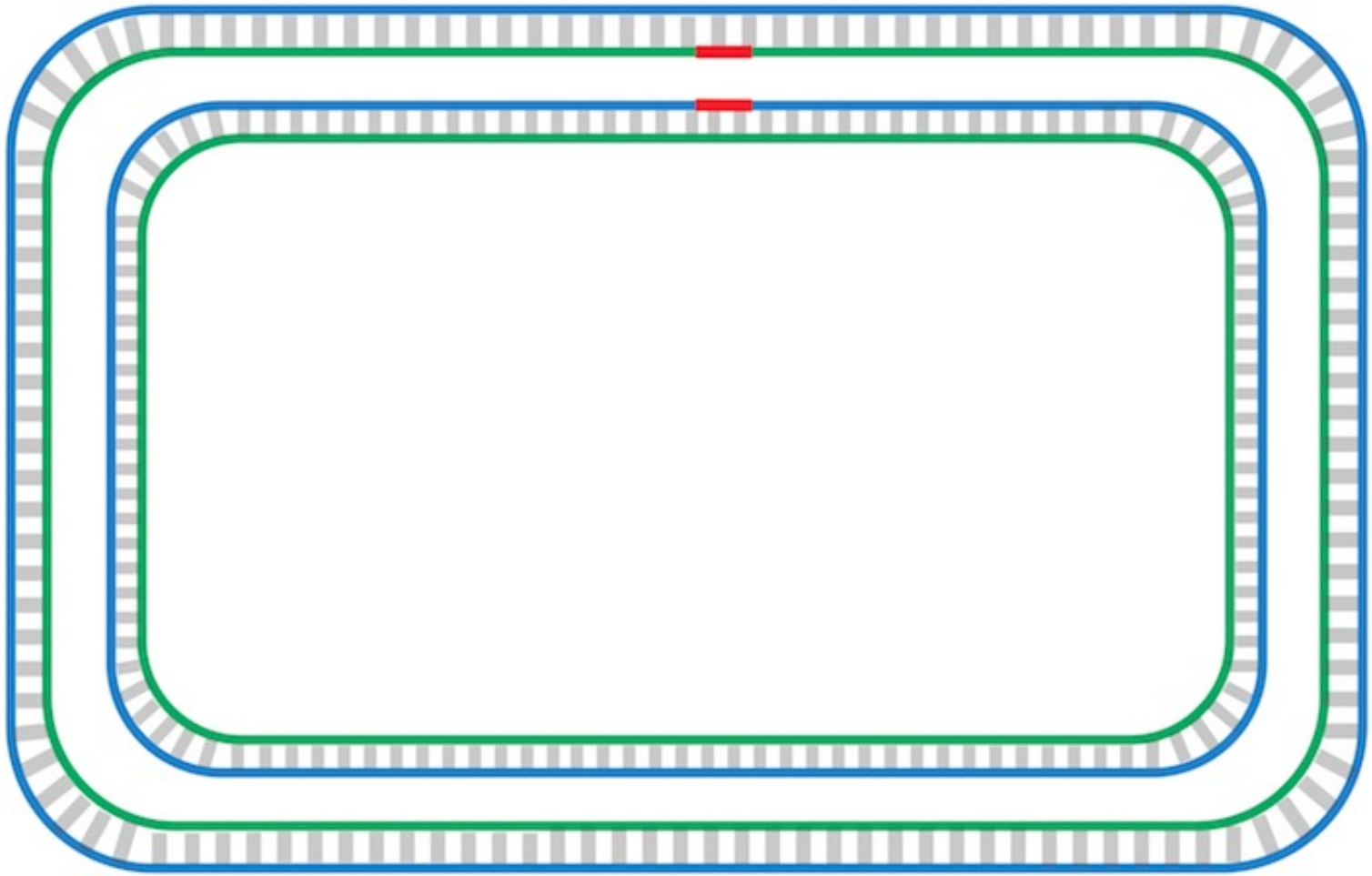
Vervollständigen der Replikation



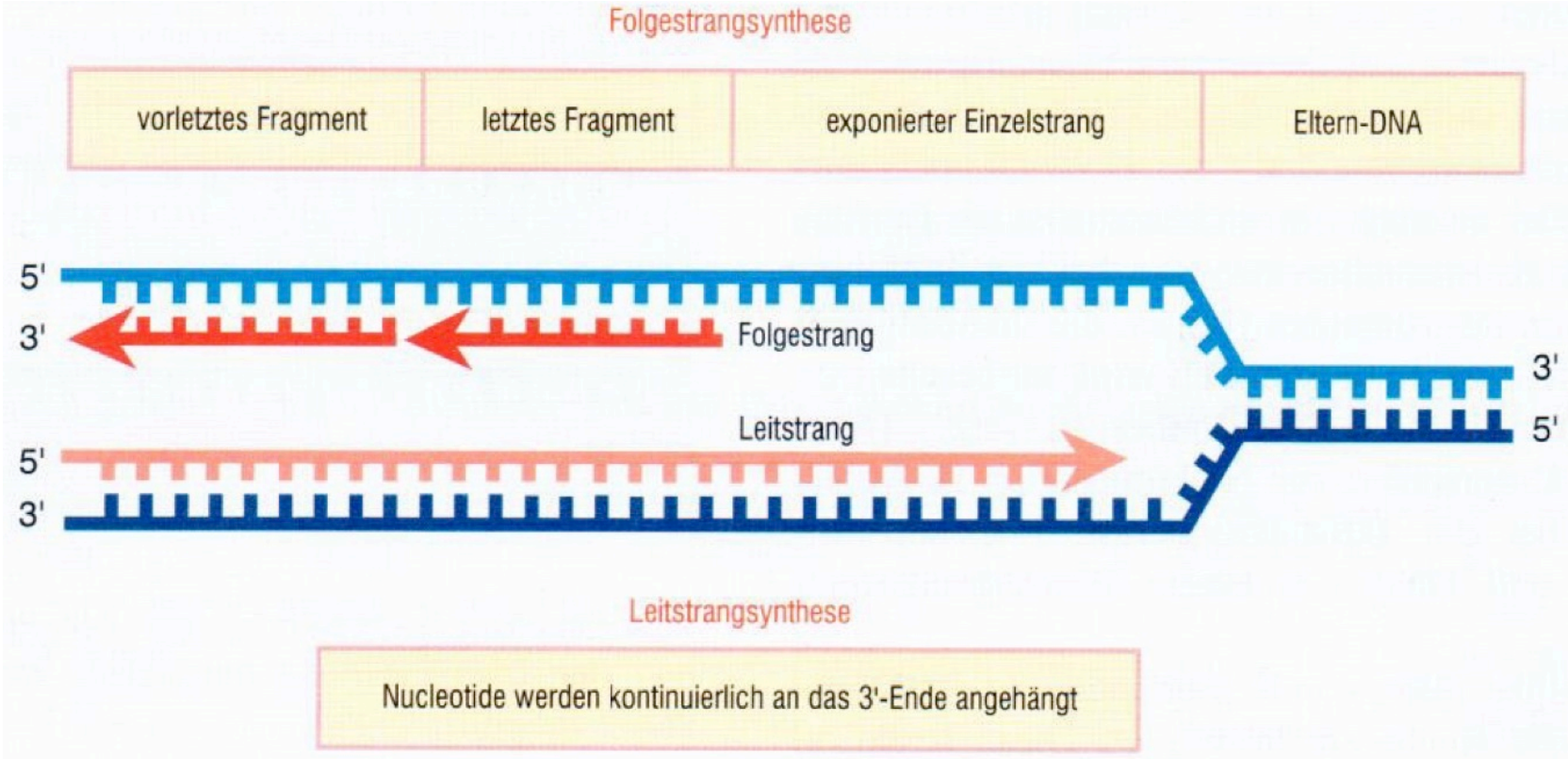
Vervollständigen der Replikation



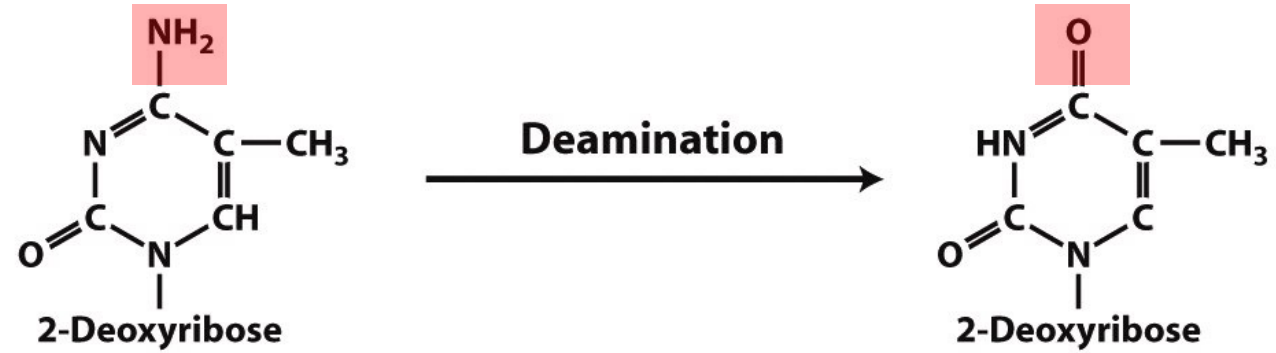
Vervollständigen der Replikation



Konsequenzen der semi-diskontinuierlichen Synthese

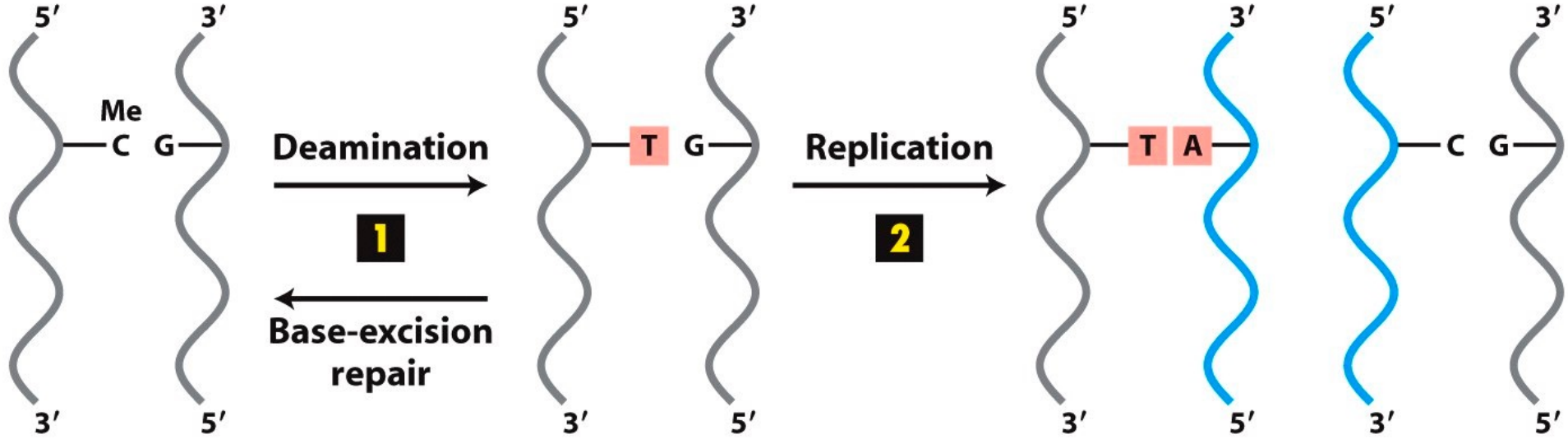


Desaminierung von ssDNA

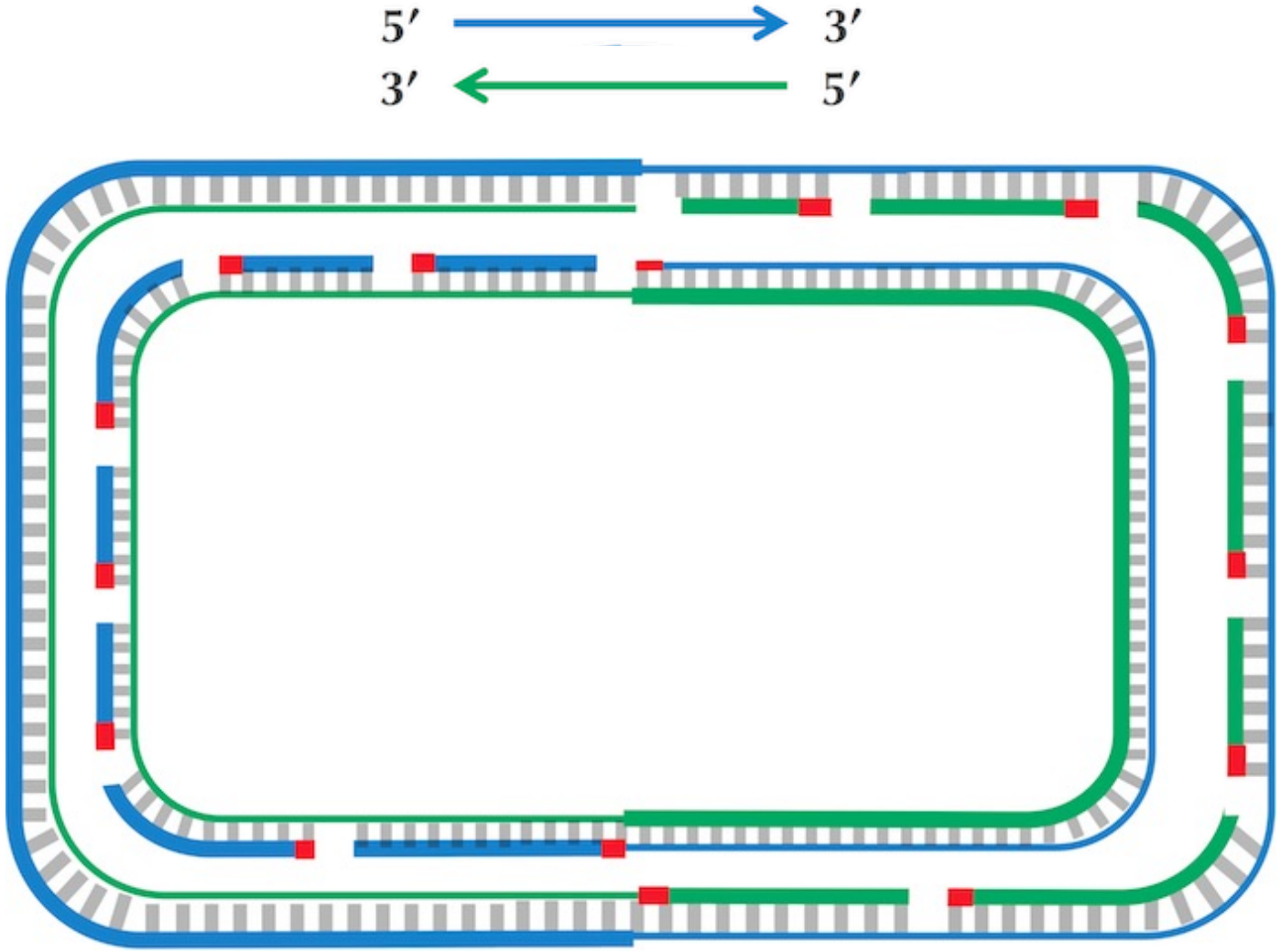


5-Methylcytosine
(“modifiziertes **C**”)

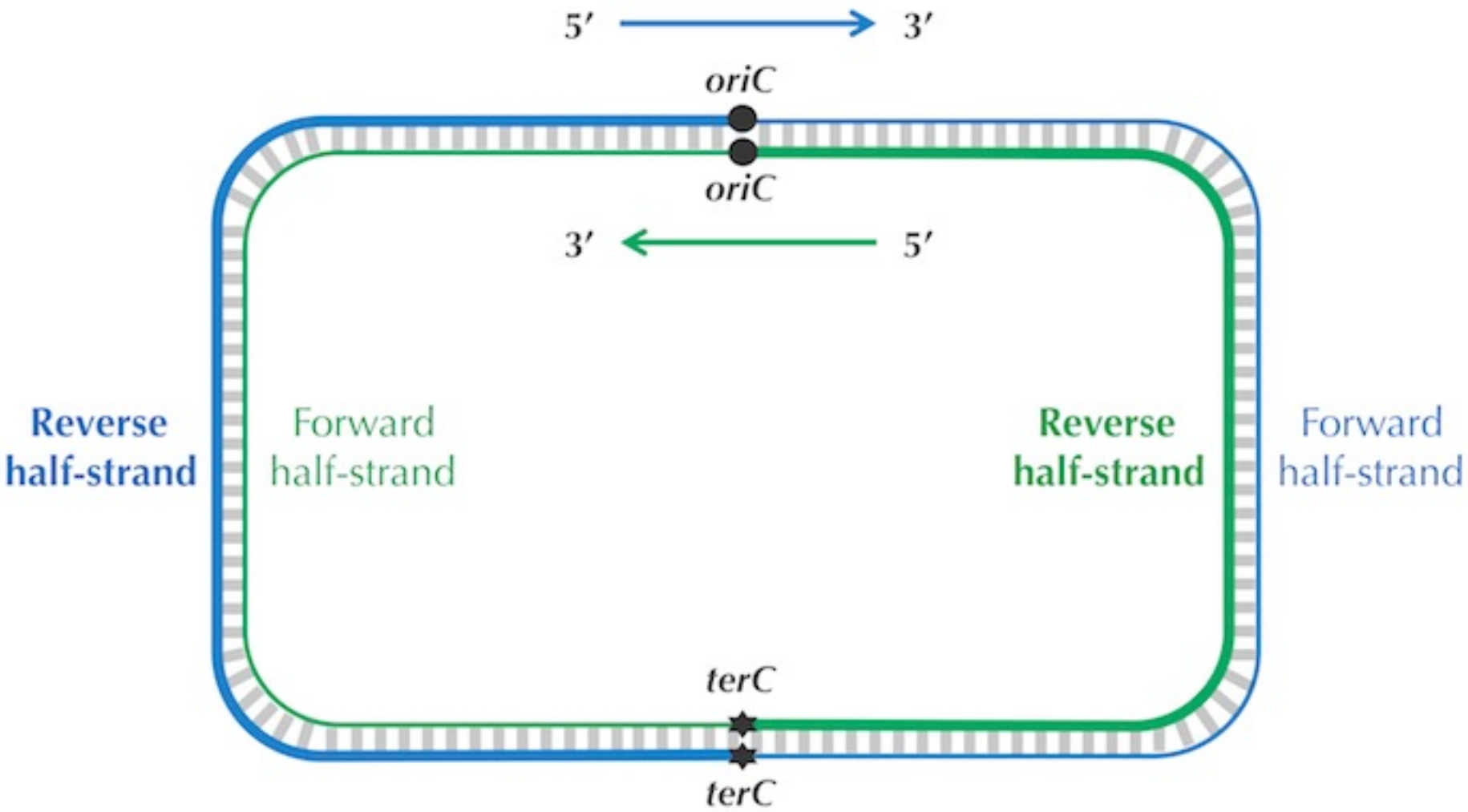
Thymine
(**T**)



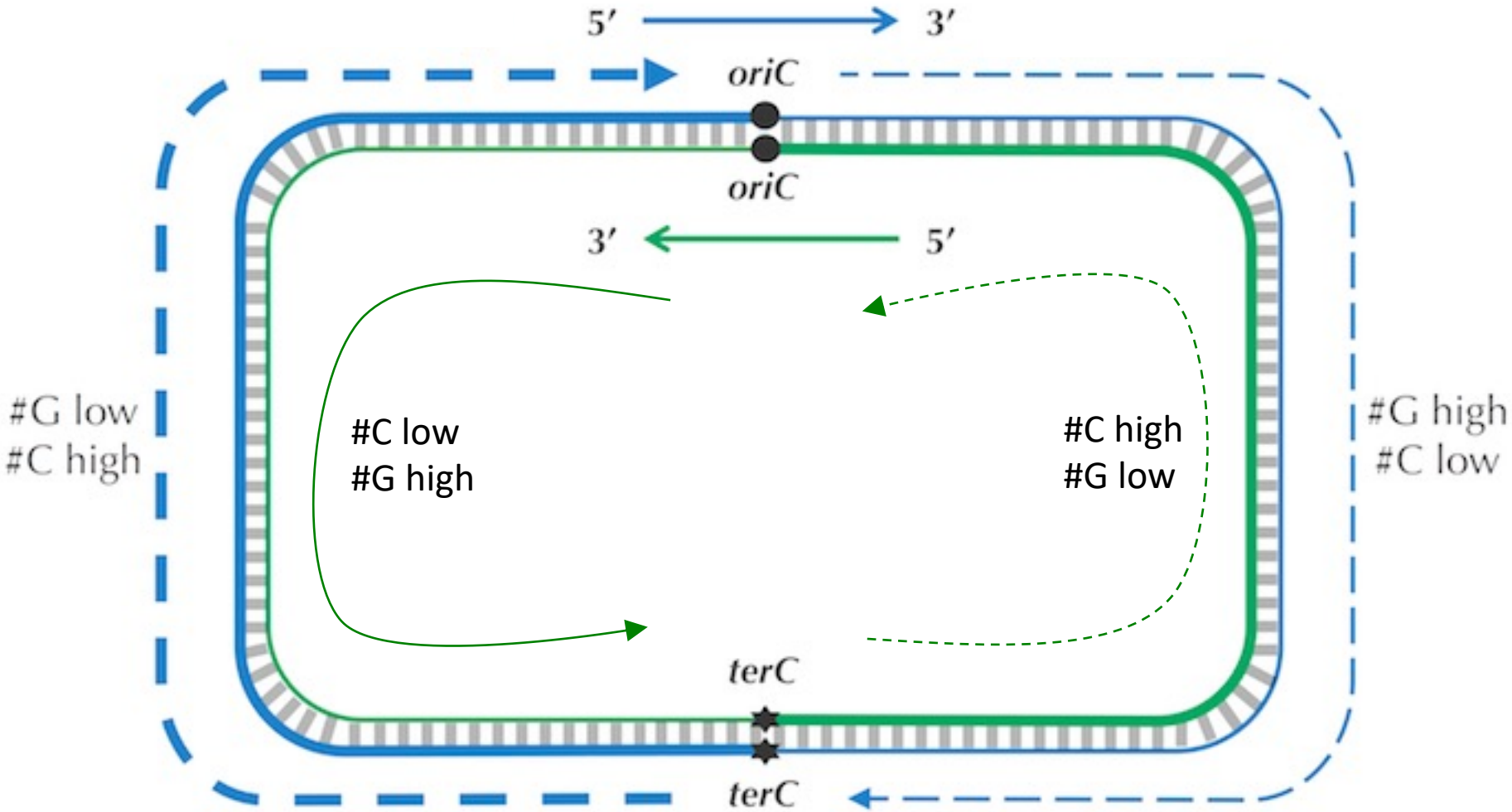
Auf den Spuren der semi-diskontinuierlichen DNA Synthese



Auf den Spuren der semi-diskontinuierlichen DNA Synthese

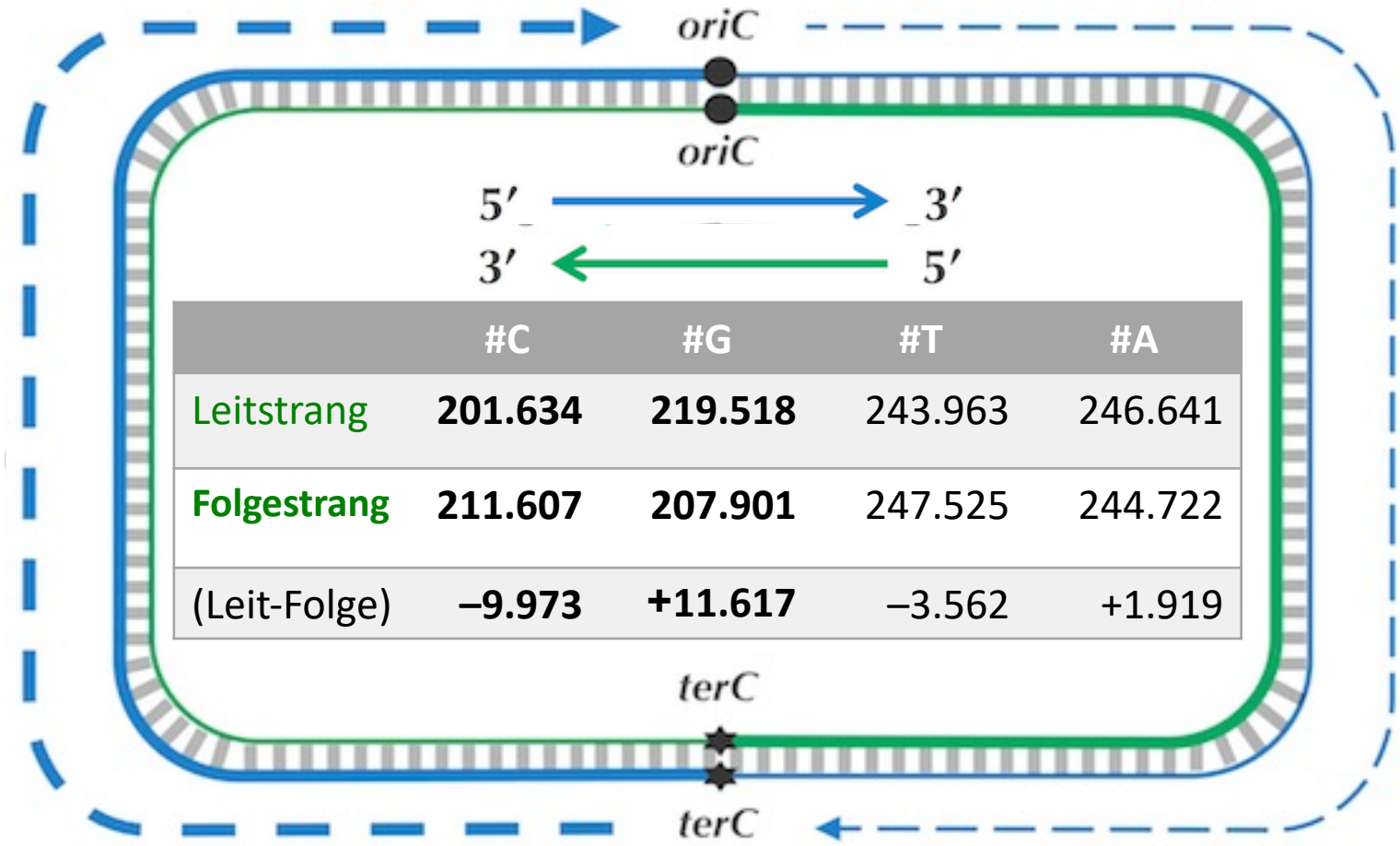


Auf den Spuren der diskontinuierlichen DNA Synthese



Thermotoga petrophila

#G = 201.634
#C = 219.518

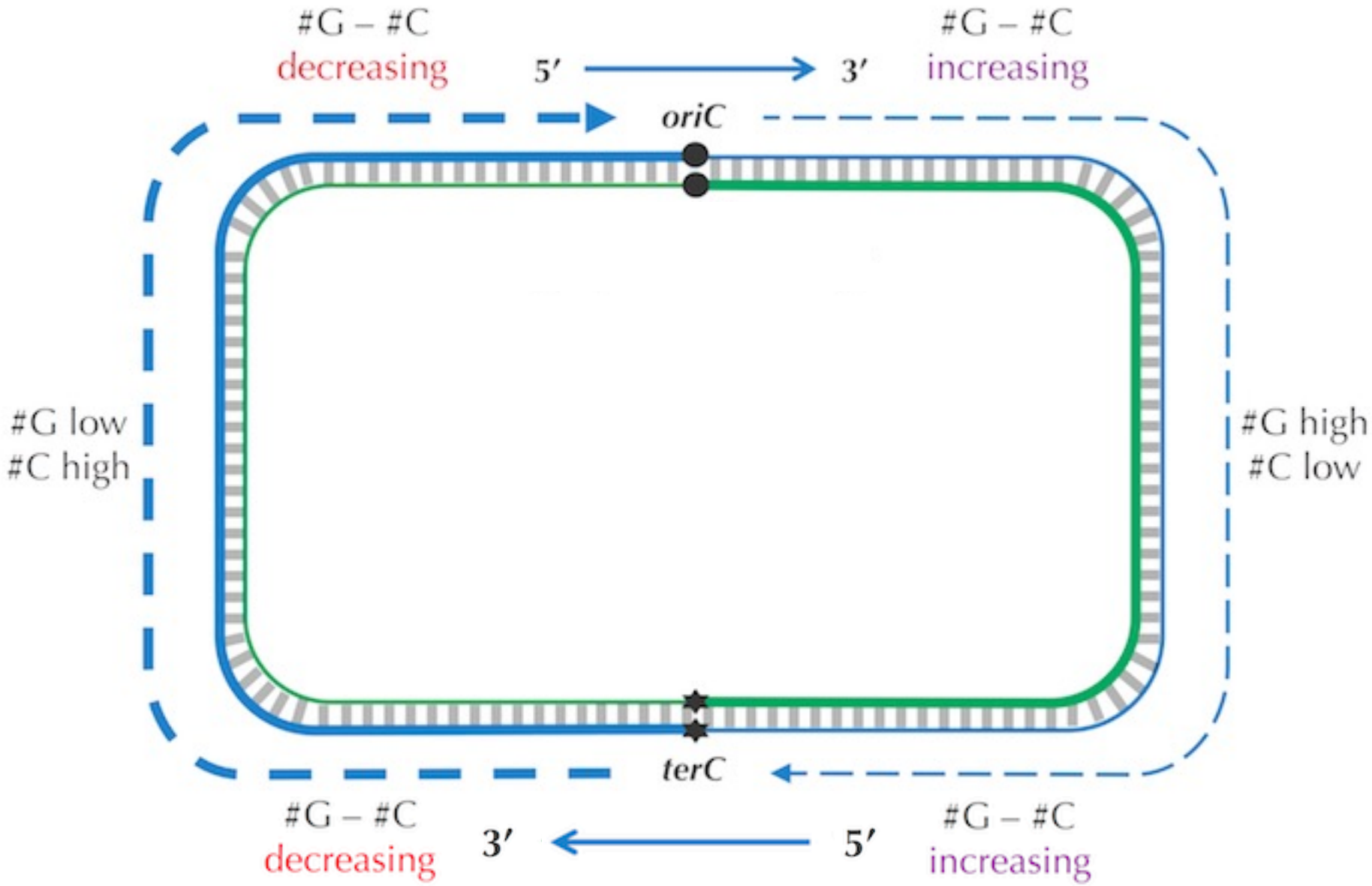


	#C	#G	#T	#A
Leitstrang	**201.634**	**219.518**	243.963	246.641
Folgestrang	**211.607**	**207.901**	247.525	244.722
(Leit-Folge)	**–9.973**	**+11.617**	–3.562	+1.919

#G = 211.607
#C = 207.901

	#C	#G	#T	#A
Leitstrang	**207.901**	**211.607**	244.722	247.525
Folgestrang	**219.518**	**201.634**	246.641	243.963
(Leit – **Folge**)	**–11.617**	**+9.973**	–1.919	+3.562

Auf den Spuren der diskontinuierlichen DNA Synthese



Das Ungleichgewichts-Diagramm

Definition:

- *Ungleichgewicht*$_i$(*Genom*): Differenz (#G – #C) an Position *i*

mit

- *Genom* = String
- *i* auf dem Interval [0 ; |Genom|-1]

(zirkuläre Genome -> Linearisierung an beliebiger Stelle)

Rekursive Berechnung:

- *Ungleichgewicht*$_0$(*Genom*) = 0
- *Ungleichgewicht*$_{i+1}$(*Genom*) = *Ungleichgewicht*$_i$(*Genom*) +

+1 wenn *Genom*$_i$= 'G'

–1 wenn *Genom*$_i$= 'C'

0 in jedem anderen Fall

Graphische Darstellung

Beispiel: *Genom* = **C**AT**GGG**CAT**CGG**CCATA**CG**CC

$Ungleichgewicht_{i+1}(Genom) = Ungleichgewicht_i(Genom) +$

+1 wenn $Genom_i$ = '**G**'

–1 wenn $Genom_i$ = '**C**'

0 in jedem anderen Fall

$Ungleichgewicht_1(Genom) = 0$

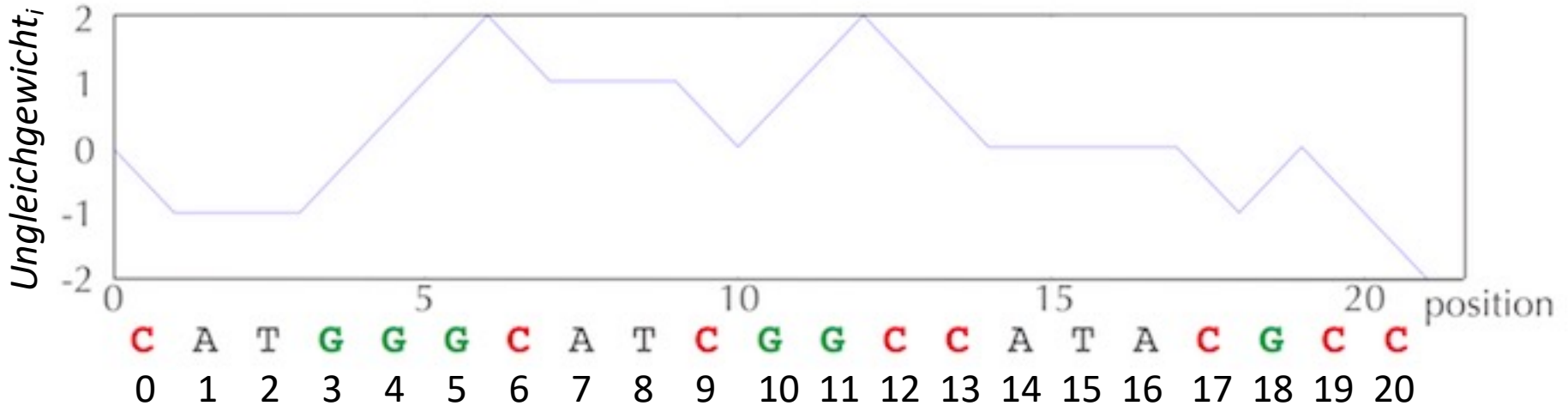
$Ungleichgewicht_1(Genom) = 0 –1$

$Ungleichgewicht_2(Genom) = –1+0$

...

$Ungleichgewicht_3(Genom) = –1+1$

...



Visualisierungen in Python

https://matplotlib.org → API docs

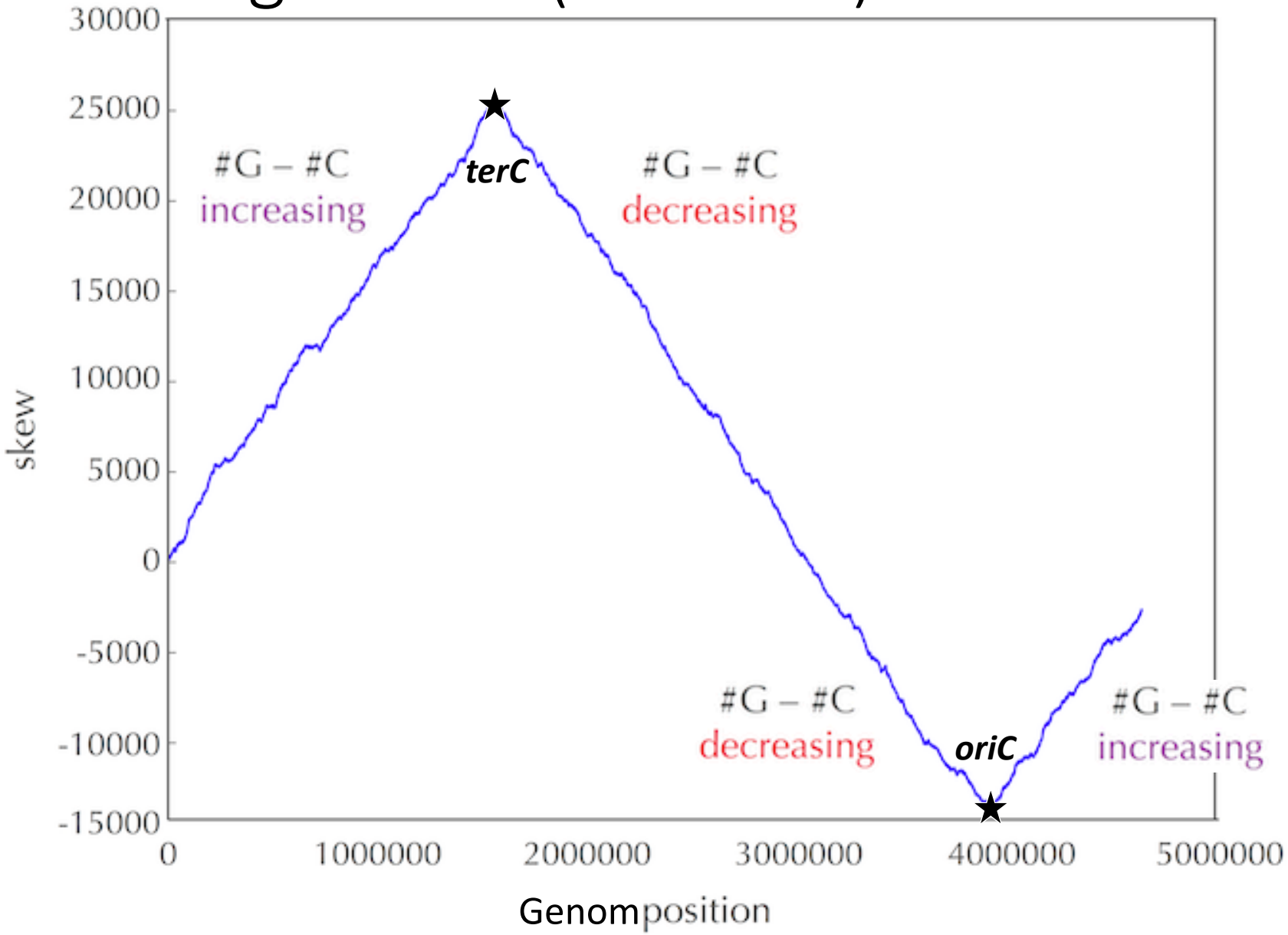
installieren mit “pip install ...”

importieren mit “import ... [as ...]” oder “from ... import ... [as ...]”

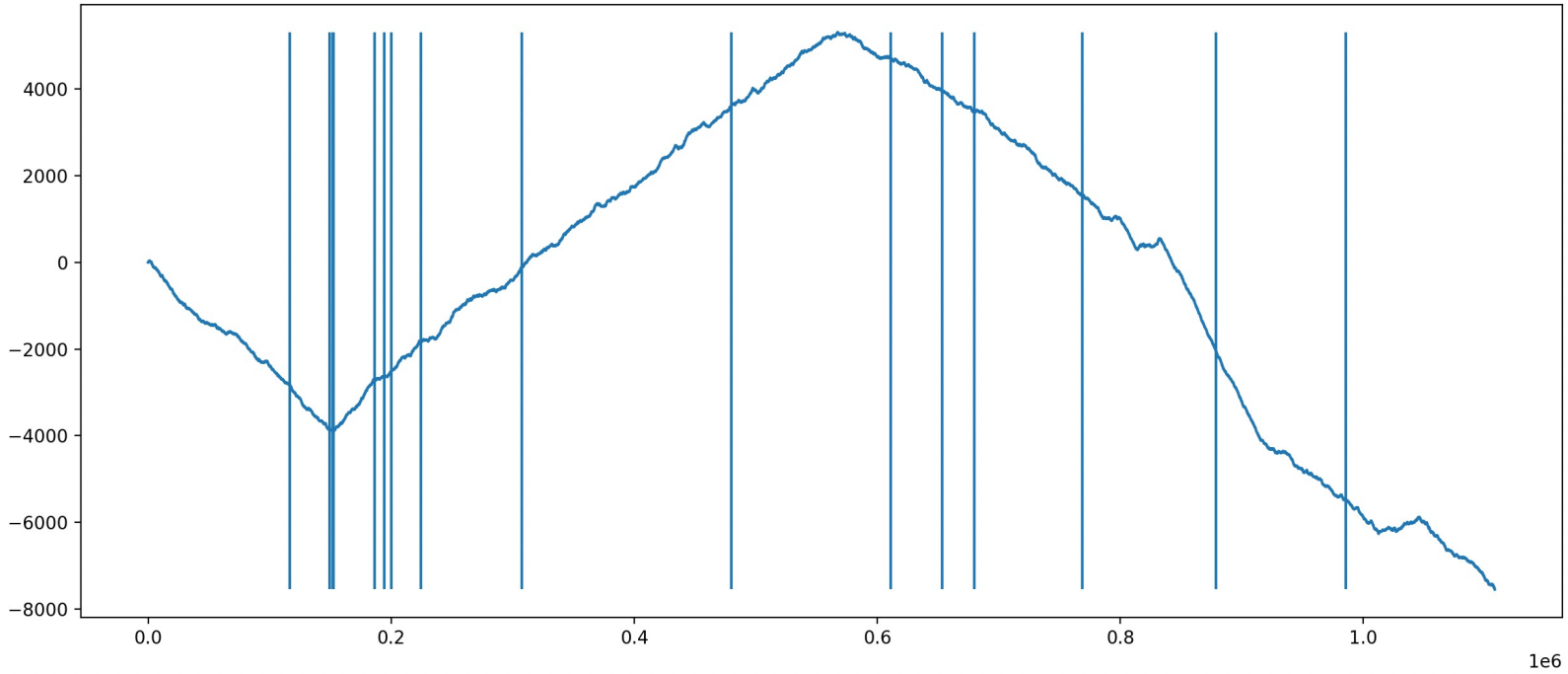
matplotlib.pyplot.scatter

```
import matplotlib.pyplot as plt
plt.scatter([1,2,3],[1,2,3])
plt.vlines([140000,170000], ymin= -8000, ymax= 5000, colors=["red","green"])
```

Das Ungleichgewichts-Diagramm weist den Weg zu *oriC* (und *terC*)



Kombiniere Info vom Ungleichgewichts-Diagramm mit Pattern-Matching



PatternMatching(ATGATCCAG, *Genom*$_{Vibrio\text{-}cholerae}$) ergibt:
116556, 149355, **151913, 152013, 152394**, 186189, 194276, 200076, 224527, 307692, 479770, 610980, 653338, 679985, 768828, 878903, 985368.

Klumpen Finden Problem

Zusammenfassung:

- ein bakterieller *oriC* ist ca. 500 Nt lang
- die "versteckten Nachrichten" im oriC sind 9-mere (*DnaA-Boxes*)
- diese *DnaA* Motive sind spezies-spezifische Muster, d.h. die Strings unterscheiden sich von Spezies zu Spezies
- *DnaA-Boxen* bilden Klumpen spezifisch im Bereich des oriCs

➡ Um den *oriC* zu lokalisieren müssen wir nach einem “Klumpen” von Nonameren in einer spezifischen Region des Genoms suchen.

Definition: "Klumpen"
Ein *k*-mer bildet einen (*L*,*t*)-Klumpen auf einem langen String *Genom*, wenn es in einem Fenster der Länge *L* mindestens *t* Male vorkommt.

Beispiel: **TGCA** bildte einen (25,3)-Klumpen in folgendem Genom
`gatcagcataagggtccCTGCAATGCATGACAAGCCTGCAGTtgttttac`

Klumpen Finden Problem: *Finde Muster der Länge k, die in einem String (L,t)-Klumpen bilden.*

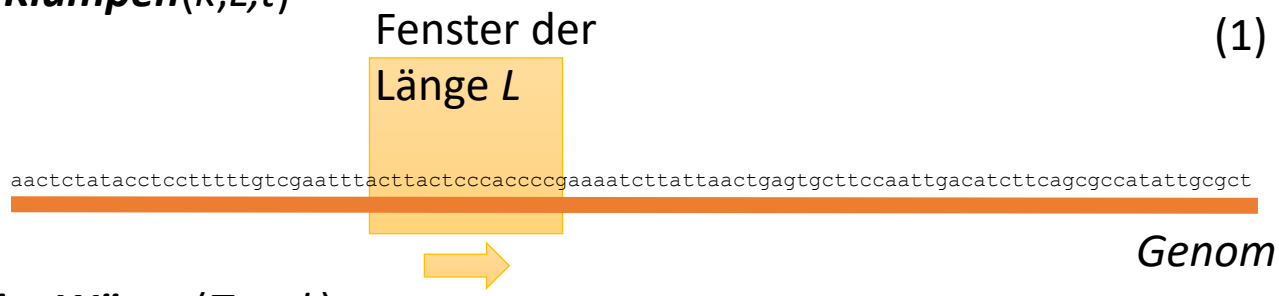
Input: Ein String *Genom*, und ganze Zahlen *k*, *L*, und *t*.

Output: Alle unterschiedlichen *k*-mere, die (*L*,*t*)-Klumpen in *Genom* bilden.

Verschachtelung mit vorigen Algorithmen

Anwendung von *HäufigeWörter*(*Text*,*k*) auf *Genom* (mit ca. 10^6 Nukleotiden!)

- ***FindeKlumpen***(*k*,*L*,*t*)



(1) Für Fenster der Länge *L* in *Genom* als *Text*

- ***HäufigeWörter***(*Text*,*k*)

(2) Für jedes *k*-mer in *Text* als *Muster*

- ***Zähle***(*Text*,*Muster*)

(3) Zähle alle *Muster* in *Text*

Algorithmus: *KlumpenFinden*()

Algorithm 1: KlumpenFinden($Genom$,k, t, L)

```
KlumpenWörter ← ∅
for i ← 0 to |Genom| − L do
    for j ← 0 to 4^k − 1 do
        Anzahl(j) ← 0;
    Text ← Genom(i, L)
    for j ← 0 to |Text| − k do
        kmer ← Text(j, k)
        index ← kmerZuIndex(kmer)
        Anzahl(index) ← Anzahl(index) + 1
        if Anzahl(index) ≥ t then
            KlumpenWörter ← KlumpenWörter ∪ {kmer}
return KlumpenWörter
```

Aufwandsabschätzung für *KlumpenFinden*()

Algorithm 1: KLUMPENFINDEN($Genom, k, t, L$)

```
KlumpenWörter ← ∅
for i ← 0 to |Genom| − L do
    for j ← 0 to 4^k − 1 do
        Anzahl(j) ← 0;
    Text ← Genom(i, L)
    for j ← 0 to |Text| − k do
        kmer ← Text(j, k)
        index ← kmerZuIndex(kmer)
        Anzahl(index) ← Anzahl(index) + 1
        if Anzahl(index) ≥ t then
            KlumpenWörter ← KlumpenWörter ∪ {kmer}
return KlumpenWörter
```

|*Genom*| - *L* + 1 Schritte ∈ **|Genom|** f. *L* << |*Genom*|

4^k Schritte

***L* - *k* + 1** Schritte ∈ **O(*L*)** f. *k* << *L*

k Schritte

KlumpenFinden(*Genom*,*k*,*t*,*L*) ∈ $O(\ |Genom| \cdot (\ 4^k + L \cdot k\ )\ )$ iff *k* << *L* << |*Genom*|