Algorithmen der Sequenzanalyse: verrauschte Motive

AlgSeq – 28/10/2024

Prof. M. Sammeth

Inexakte Vorkommnisse eines Musters

Motive mit einer Abweichung (im *oriC* von *Vibrio cholerae)*:

atca**ATGATCAAC**gtaagcttctaagc**ATGATCAAG**gtgctcacacagtttatccacaac
ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca
cggaaag**ATGATCAAG**agaggatgatttcttggccatatcgcaatgaatacttgtgactt
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt
acgaaag**CATGATCAT**ggctgttgttctgtttatcttgttttgactgagacttgttagga
tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat
tgataatgaatttacatgcttccgcgacgatttacct**CTTGATCAT**cgatccgattgaag
atcttcaattgttaattctcttgcctcgactcatagccatgatgagct**CTTGATCAT**gtt
tccttaaccctctatttttacggaaga**ATGATCAAG**ctgctgct**CTTGATCAT**cgtttc

ATGATCAAC und **CATGATCAT**

Acht (statt sechs) Vorkommnisse eines Nonameres!

$$p_f(N,k,t) \approx \frac{\binom{N-t(k-1)}{t}}{4^{(t-1)}\,k}$$

p_f(540, **9**, **8**)= 7.242252695405615**e-22**

k	*t*	p_f(540, 9, *t*)
7	9	7.4515173163413 52e-16
8	7	1.6504527748087858e-14
9	6	1.5433634500541 06e-14
10	4	2.3042467465345107e-09
11	4	3.4871074194972 77e-11
12	3	7.6711206276058 87e-08

Mismatches und die Hamming Distanz

Vergleich zweier *k*-mere *p* und *q*

p:	p_1	p_2	p_3	...	p_k
q:	q_1	q_2	q_3	...	q_k

$p_i \neq q_i$ → ein Mismatch

HAMMINGDISTANZ(p, q) := Anzahl aller Mismatches über alle k Positionen

HAMMINGDISTANZ(p, q) $\leq d$ ⟷ höchstens d Mismatches im Vergleich p mit q

Zählen$_d$(*Text*, *Muster*) := Anzahl der Vorkommnisse von *Muster* in *Text* mit höchstens d Mismatches.

Bsp.:

Zählen$_1$(**AA**C**AA**GC**AT**A**AA**C**A**TT**AAA**G**A**G, **AAAAA**) = 4, d.h. {**AA**C**AA**, **AT**A**AA**,

Inexaktes Zählen und *Häufige Wörter* mit *Mismatches*

Algorithm: ZÄHLEN($Text$, $Muster$,d)

```
Zähler ← 0
for i ← 0 bis |Text| − |Muster| do
    Wort ← Text(i, |Muster|)
    if HammingDistanz(Wort, Muster) ≤ d then
        Zähler ← Zähler + 1
return Zähler
```

Häufige Wörter mit RC und Mismatches Problem: Finde das/die häufigsten *k*-mer(e) die (zusammen mit dem reversen Komplement und mit höchstens *d* Mismatches) in einer Zeichenkette vorkommen.

Eingabe: Ein String *Text*, ganze Zahlen *k* und *d*.

Ausgabe: Alle *k*-mere *Muster*, die $\text{ZÄHLEN}_d(Text, Muster) + \text{ZÄHLEN}_d(Text, Muster')$ ihres reversen Komplements Muster' maximieren (über alle *k*-mere in Text).

Unmittelbare Nachbarn ($d \leq 1$)

Unmittelbare Nachbarn := Zwei Zeichenketten *p* und *q*, die sich in höchstens einem Mismatch voneinander unterscheiden (HAMMINGDISTANZ(p, q) ≤ 1)

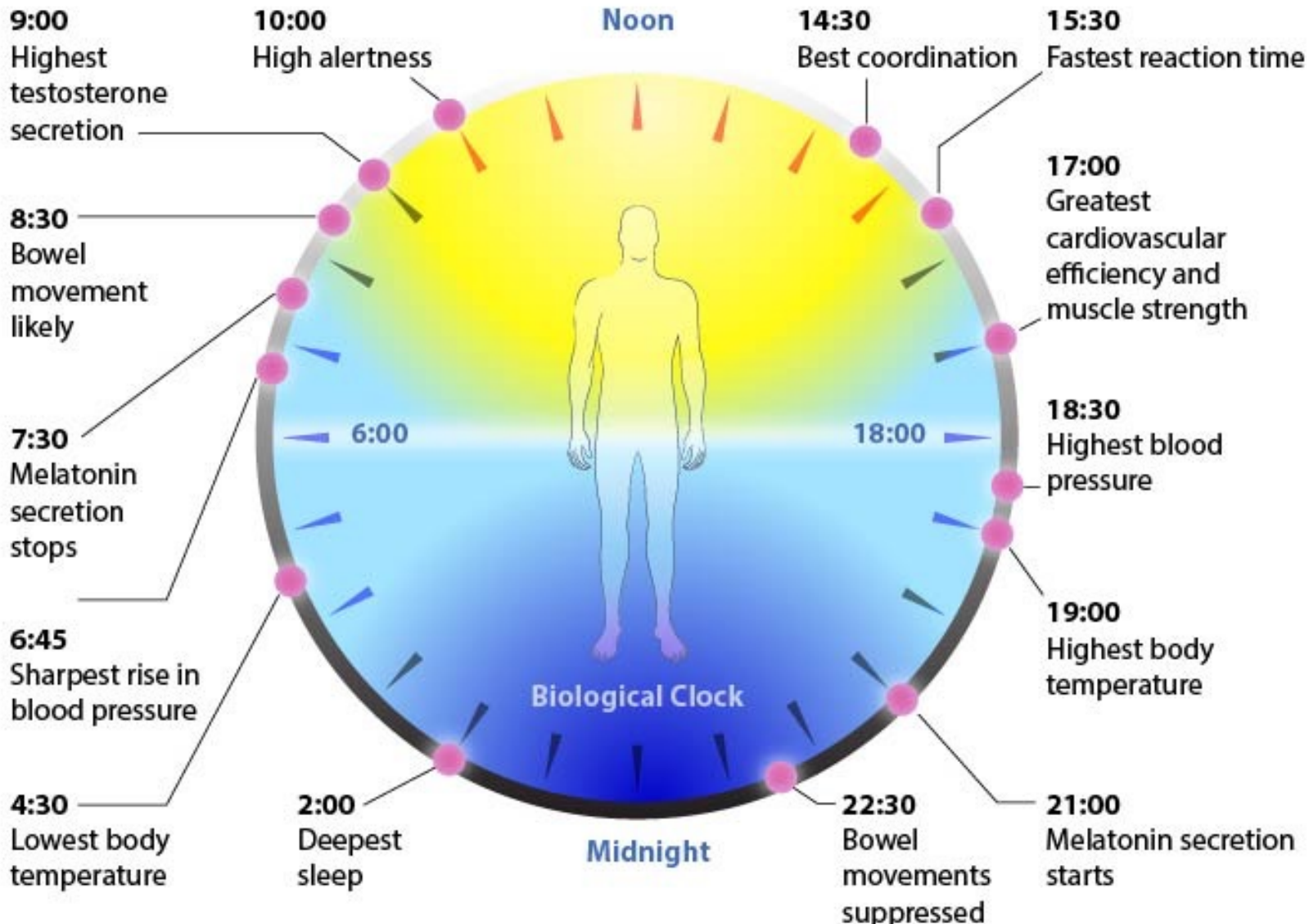
Algorithm: UNMITTELBARENACHBARN($Muster$)

```
Nachbarschaft ← {Muster}
for i = 1 to |Muster| do
    Symbol ← i-tes Nukleotid von Muster
    for jedes Nukleotid x unterschiedlich von Symbol do
        Nachbar ← Muster mit dem i-ten Nukleotid substituiert durch x
        Nachbarschaft ← Nachbarschaft ∪ {Nachbar}
return Nachbarschaft
```

Häufige Wörter mit RC und Unmittelbaren Nachbarn Problem:
Finde das/die häufigsten *k*-mere mit höchstens einem Mismatch ....

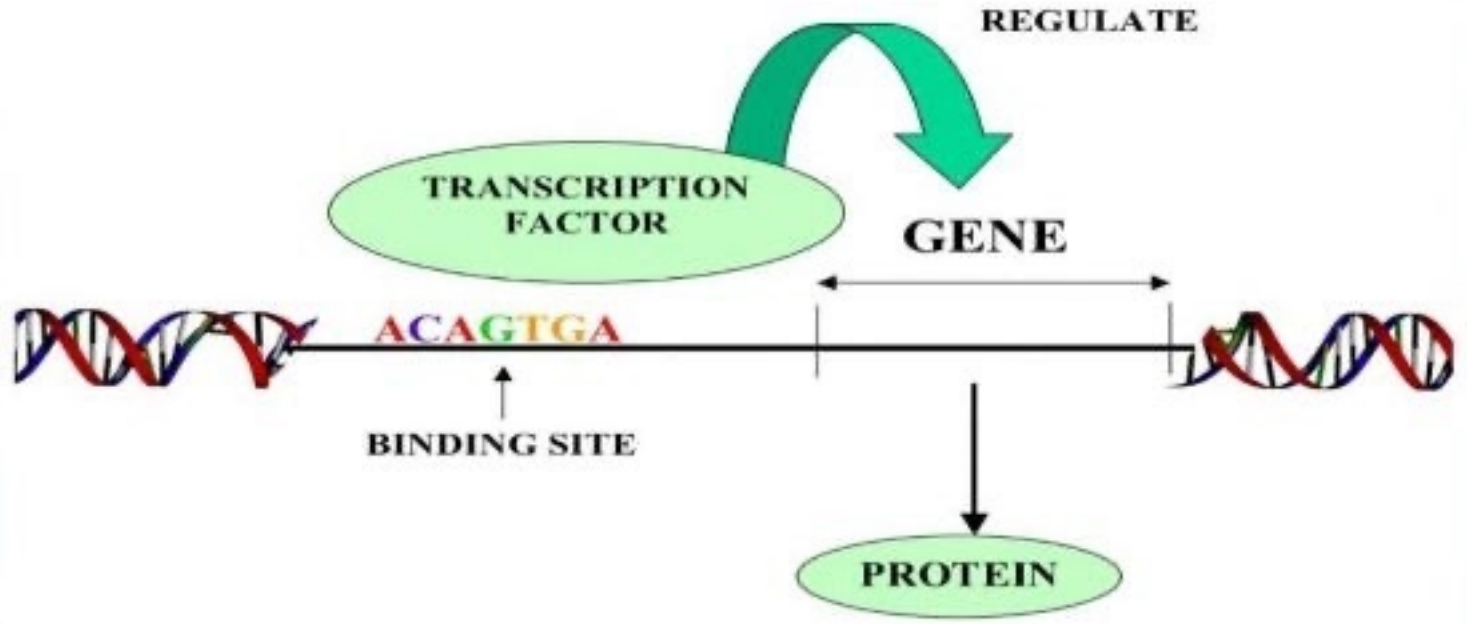
Transkriptionsfaktoren, z.B. für “Molekulare Uhren”

Gibt es ein “Uhr-Gen”?



Transkriptionsfaktoren und ihre Bindestellen

Transkriptionsfaktoren sind Proteine, die Bindestellen in der DNA erkennen und benachtbarte Gene an-/ab-schalten können.



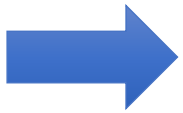
- **mehrere Gene** (mit ähnlichen Funktionen) werden durch den/die gleiche(n) **Transkriptionsfaktor**(en) kontrolliert

- **Motiv** (“regulatorisches Motiv”) der Bindestellen des **gleichen** Transkriptionsfaktors ist oft **inexakt**.

z. B. NF-$\varkappa$B (Transkriptionsfaktor)

NF-$\varkappa$B kontroliert Gene der Immunantwort in der Fruchtfliege.

1	TCGGGGgTTTtt
2	cCGGtGAcTTaC
3	aCGGGGATTTtC
4	TtGGGGAcTTtt
5	TCGGGGATTTCC
6	TtGGGGAcTTCC
7	TCGGGGATTcat
8	TCGGGGATTcCt
9	TaGGGGAacTaC
10	TCGGGtATaaCC



Motive von 10 Binstellen für NF-$\varkappa$B,
inexaktes (nicht konserviertes) Motiv

Verstecken Spielen mit Motiven

(1) 15-mer Motiv **AAAAAAAAGGGGGGG** (exakt) implantiert in 10 Sequenzen:

1 atgaccgggatactgat**AAAAAAAAGGGGGGG**ggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg
2 acccctatttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaata**AAAAAAAAGGGGGGG**a
3 tgagtatccctgggatgactt**AAAAAAAAGGGGGGG**tgctctcccgatttttgaatatgtaggatcattcgccagggtccga
4 gctgagaattggatg**AAAAAAAAGGGGGGG**tccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga
5 tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaat**AAAAAAAAGGGGGGG**cttatag
6 gtcaatcatgttcttgtgaatggattt**AAAAAAAAGGGGGGG**gaccgcttggcgcacccaaattcagtgtgggcgagcgcaa
7 cggttttggcccttgttagaggcccccgt**AAAAAAAAGGGGGGG**caattatgagagagctaatctatcgcgtgcgtgttcat
8 aacttgagtt**AAAAAAAAGGGGGGG**ctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta
9 ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcat**AAAAAAAAGGGGGGG**accgaaagggaag
10 ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagctt**AAAAAAAAGGGGGGG**a

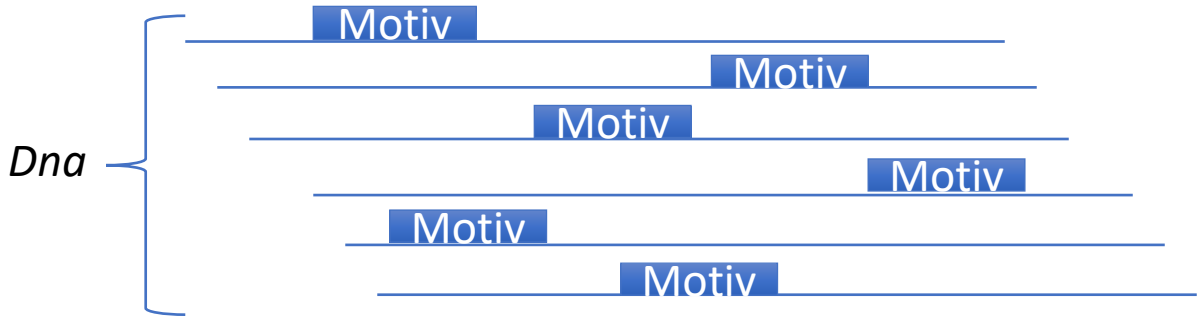
(2) 15-mer Motiv **AAAAAAAAGGGGGGG** mit *d*=4 Mismatchen implantiert in 10 Sequenzen:

1 atgaccgggatactgat**AgAAgAAAGGttGGG**ggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg
2 acccctatttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaata**cAAtAAAAcGGcGGG**a
3 tgagtatccctgggatgactt**AAAAtAAtGGaGtGG**tgctctcccgatttttgaatatgtaggatcattcgccagggtccga
4 gctgagaattggatg**cAAAAAAAGGGattG**tccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga
5 tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaat**AtAAtAAAGGaaGGG**cttatag
6 gtcaatcatgttcttgtgaatggattt**AAcAAtAAGGGctGG**gaccgcttggcgcacccaaattcagtgtgggcgagcgcaa
7 cggttttggcccttgttagaggcccccgt**AtAAAcAAGGaGGGc**caattatgagagagctaatctatcgcgtgcgtgttcat
8 aacttgagtt**AAAAAAtAGGGaGcc**ctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta
9 ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcat**ActAAAAAGGaGcGG**accgaaagggaag
10 ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagctt**ActAAAAAGGaGcGG**a

→ (1) lösbar, aber (2) unlösbar mit Algorithmen für HäufigeWörter

Problem Formulierung

(*k,d*)-Motiv := Sei *Dna* eine Menge von Strings und *d* ein Integer, dann ist ein *k*-mer ein **(*k,d*)-Motiv**, wenn es in jedem String aus *Dna* mit höchstens *d* Mismatches vorkommt.



Implantiertes Motiv Problem:
Finde alle (k,d)-Motive in einer Menge von Zeichenketten.

Input: Eine Menge von Zeichenketten Dna, Integer *k* und *d*.

Output: Alle (*k,d*)-Motive in *Dna*.

Implantiertes Motiv mit roher Gewalt lösen

hier betrachtete Instanz des Problems (NF-$\varkappa$B Bindemotive): **Subtiles Motif Problem**

Das 15-mer Motiv **AAAAAAAAGGGGGGG** *wurde mit vier zufälligen Mutationen implantiert in zehn 600nt (=typische Länge von upstream regulatorischen Regionen) Sequenzen.*

1 atgaccgggatactgat**AgAAAgAAGGttGGG**ggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg
2 acccctatttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaata**cAAtAAAAcGGcGGG**a
3 tgagtatccctgggatgactt**AAAAtAAtGGaGtGG**tgctctcccgattttgaatatgtaggatcattcgccagggtccga
4 gctgagaattggatg**AAAAAAAAGGGattG**tccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga
5 tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaat**AtAAtAAAGGaaGGG**cttatag
6 gtcaatcatgttcttgtgaatggattt**AAcAAtAAGGGctGG**gaccgcttggcgcacccaaattcagtgtgggcgagcgcaa
7 cggttttggcccttgttagaggcccccgt**AtAAAcAAGGaGGGc**caattatgagagagctaatctatcgcgtgcgtgttcat
8 aacttgagtt**AAAAAAtAGGGaGcc**ctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta
9 ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcat**ActAAAAAGGaGcGG**accgaaagggaag
10 ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagctt**ActAAAAAGGaGcGG**a

Brute Force (*Rohe Gewalt*) := Algorithmische Vorgehensweise, bei der alle möglichen Lösungen geprüft werden, und dann die beste Lösung vorgeschlagen wird.
(auch: exhaustive Suche)

wie lautet ein “brute force” Ansatz für das ***Implantiertes Motiv*** Problem?

Implantiertes Motiv mit roher Gewalt lösen

ein “brute force” Ansatz für das ***Implantiertes Motiv*** Problem:

Algorithm: MOTIVEAUFZAEHLEN(Dna,k, d)

```
Patterns ← ∅
for jedes k-mer Muster in Dna do
    for jedes k-mer Muster', das sich von Muster mit höchstens d
     Unterschieden (“Mismatches”) unterscheidet do
        if Muster' in jeder Zeichenkette von Dna mit höchstens d
         Mismatches vorkommt then
            Patterns ← Patterns ∪ {Muster'}
entferne alle Duplikate aus Patterns
return Patterns
```

paarweise Vergleiche zwischen Sequenzen helfen kaum (s.u.) → **zwei** Sprünge im Vergleich: *Muster* → *Muster’* → ZÄHLEN$_2$*(Muster’)*

sehr langsam für große Were von *k* und/oder *d* !

AgAAgAAAGGttGGG
|| || | ||| |
cAAtAAAAcGGGGcG