

Aufgabe 1 - Einfaches String Matching

Silke Trißl

Wissensmanagement in der Bioinformatik



Exaktes Matching

- Gegeben: P (Pattern) und T (Text), $|P| \leq |T|$
- Gesucht: Sämtliche Vorkommen von P in T

Beispiel

Auffinden der Erkennungssequenzen von Restriktionsenzymen

Eco RV - GATATC

```
tcagcttactaattaaaaatccttctagtaagtgcataagatcaagaaaaataataaaaataatggaacatggcacatcttctaaactcttcacagattgctaatga  
ttataataaagaataaatgtataatctttatggtaacggaatttctaaaaataatcaagcaccatggaatgcaaaaagaaggactctgtaattgggtact  
atccaactcaatgcaagtggaaactaagtgggtataataactctttttacataatataatgtagttatcttgggaagcgaaggacaatttcaatctgctaataaaggat  
atatttattttgtgaaataaaaaatgagaagtatggtatcagattaaacttttgagaaaggttaagtgaagtaaaagctgtatactccagcaataaagtcaaataggc  
gaaaaacttttaataacaaagttaataatcaatcttgggaattgaaatgtcaagataatcttccagataagtagtgaagtagtttaaatcttctttttgtatt  
actcaatgaaggtaacgcaacaagattagagtatataatggccaataaggtttgctgtaggaaaaatcttcaaggagatcgcgagaggctctcctcaaatctcaga  
gatggatgttttagatggtgggttaagaagaagcagattataatccagcaaaactagaccttaggttatcaagcgaggcaataagtttaattggaattgtaaaagat  
ctaattcttctcattggtggagaaaaactagttaactcttccccatgcaggccataggtcgaatcagatctgctcaagcaaaaggaatgtgagtgtagact  
ttaaaccattttataatgacttagagaatcaatgcaatctttaggttactttcttaacaatgtgacaatatttatcgcatgaagtaggtatgaaaaaggcaataat  
tattcagttacatagagattatagctggctcattcttagttatagacttttgacaagatagcttagaaaaatagattatagagcttaataaaagagaactctctggaat  
tagctgctttggtgcagctgtaaggtcattggatggctccagcttactgggttaggttttaatagaaaaatccccatgatgctaattatctctcctattgagaa  
caactgcaagatgagtgcaaaattgggtcattataactgctgggtgctatagtagttatccttagaaaagatatataatctgataaagcaaaactctggggaaaaat
```

Zeichenketten

■ Definition

Ein *String* S ist eine von links nach rechts angeordnete Liste von Zeichen eines Alphabets \mathcal{S}

- $|S|$ ist die Länge des Strings
- Positionen in S sind $1, \dots, |S|$ (wir zählen ab 1!)
- $S(i)$ beschreibt das Zeichen an der Position i im String S
- $S[i..j]$ ist der Substring, welcher an Position i beginnt und an Position j endet
- $S[i..j]$ ist ein leerer String, falls $i > j$
- $S[1..i]$ heißt **Präfix** von S bis zur Position i
- $S[i..]$ ist das **Suffix** von S , welches an Position i beginnt
- **Echte Präfixe und echte Suffixe** umfassen nicht den gesamten String S und sind nicht leer

Naiver Ansatz

1. P und T an Position 1 ausrichten
2. Vergleiche P mit T von links nach rechts
 - Zwei ungleiche Zeichen \Rightarrow Gehe zu 3
 - Zwei gleiche Zeichen
 - P noch nicht durchlaufen \Rightarrow Verschiebe Pointer auf P , gehe zu 2
 - P vollständig durchlaufen \Rightarrow Merke Vorkommen von P in T
3. Verschiebe P um ein Zeichen nach rechts
4. Wenn P noch nicht über $|T| - |P|$ hinaus, gehe zu 2

```
T   ctgagatcgcgta
P   gagatc
    gagatc
     gagatc
      gagatc
       gatatc
        gatatc
         gatatc
          gatatc
```

Naiver Ansatz (cont.)

```
for i = 1 to |T| - |P| + 1
  match := true;
  j := 1;
  while ((match) and (j <= |P|))
    if (T(i + j - 1) <> P(j)) then
      match := false;
    else
      j := j + 1;
  end while;
  if (match) then
    -> OUTPUT
end for;
```

Worst-case

T	aaaaaaaaaaaaa
P	aaaaat
	aaaaat
	aaaaat
	aaaaat

Vergleiche : $(n - 1) * (m - n + 1) \dots \Rightarrow O(m*n)$

Aufgabe 1

- Erstellt ein Java Programm mit dem
 - alle Vorkommen eines **String P** in einem zweiten **String T** gefunden werden können
- möglich:
 - **naiver Ansatz**
 - **Z-Box Algorithmus**
 - **Boyer-Moore**
 - **Knuth-Morris-Pratt**
 -

Konkrete Aufgabe



- Eingabe
 - **Pattern P** und **Text T** innerhalb der Anwendung als 'String' definieren
- Ausgabe
 - **Positionen**, an denen Pattern P in Text T exakt gefunden wird
 - **Anzahl an Vergleichen**, die nötig waren, um alle Vorkommen zu finden