Informatik II: Algorithmen und Datenstrukturen SS 2015

Vorlesung 12a, Dienstag, 14. Juli 2015

(String Matching, Teil 1)

Prof. Dr. Hannah Bast
Lehrstuhl für Algorithmen und Datenstrukturen
Institut für Informatik
Universität Freiburg

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Organisatorisches
  - Ihre Erfahrungen mit dem Ü11 (Editierdistanz)
  - Nachmeldungen für die Evaluation
- Inhalt
  - String Matching

    Grundlagen ... Definition + Beispiel

    Naiver Algorithmus ... Beispiel + Code

    Knuth-Morris-Pratt (KMP) Algorithmus ... Beispiel + Code
  - Ü12, Aufgabe 1: KMP zur Plagiatserkennung

    Noch nicht online, kommt später heute

Erfahrungen mit dem Ü11

UNI FREIBURG

- Zusammenfassung / Auszüge Stand 14. Juli 12:00
  - Pflichtaufgabe war gut machbar, insbesondere zeitlich
  - Bonusaufgabe war vergleichsweise tricky

    Haben einige probiert aber dann doch gelassen

    Nochmal: die Zusatzaufgabe waren für die gedacht, denen es ansonsten zu leicht ist … nicht zum Punktesammeln
  - ED("AAEBEAABEAREEAEBA", "RBEAAEEBAAAEBBAEAE") ist selbstverständlich 11
  - Edi-Tier muss wahlweise verhungern oder die leere Zeichenkette rächt ihre nicht-leeren Geschwister

Grundlagen 1/3

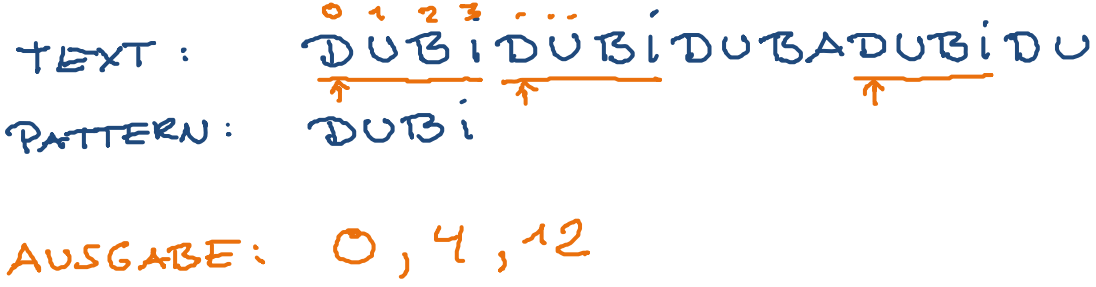
UNI FREIBURG

- Definition
  - Gegeben zwei Zeichenketten / strings:

    Ein Text (engl. text) typischerweise **lang**

    Ein Muster (engl. pattern) typischerweise **kurz**
  - Finde alle Vorkommen des Musters im Text

    Es werden die Anfangspositionen zurückgegeben
  - Notation: wir benutzen durchgängig n für die Länge des Textes und m für die Länge des Musters



Grundlagen 2/3

UNI FREIBURG

- Motivation
  - Jeder Editor hat eine "Find" Funktion (Strg+F)
  - Jede Programmiersprache hat Methoden dafür

    Python: find(text, pattern, start, end)

    Java: String.indexOf(pattern, start)

    C++: std::string.find(pattern, start)

    Damit bekommt man das nächste Vorkommen ab einer bestimmten Position (start)

    Durch wiederholtes Aufrufen dann alle Vorkommen

Grundlagen 3/3

UNI FREIBURG

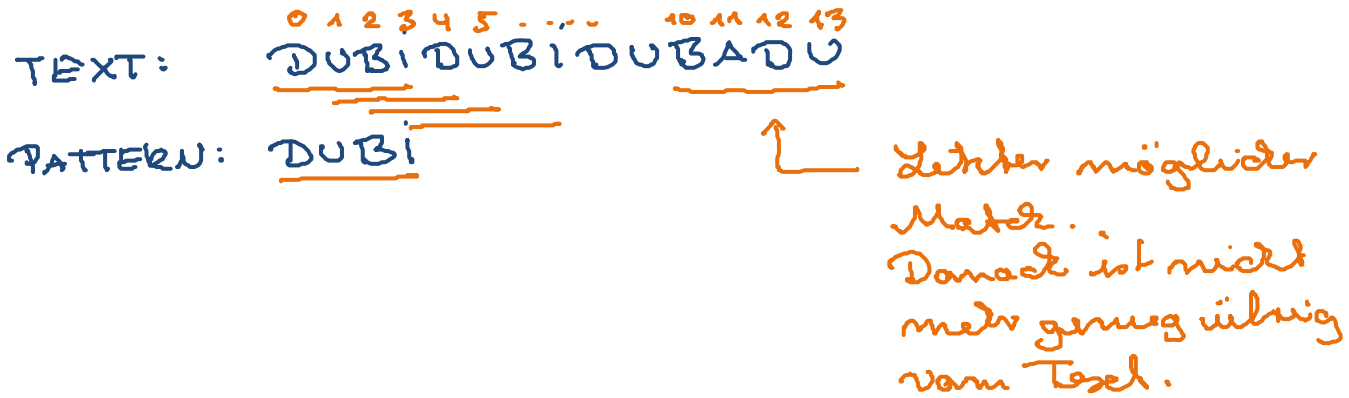
- Mehr Motivation
  - Auch zentral für die Plagiatserkennung
  - Da hat man allerdings typischerweise nicht nur ein Pattern, sondern eine Menge davon, die man wiedererkennen will

    Dazu mehr in der Vorlesung morgen

Naiver Algorithmus 1/2

UNI FREIBURG

- Prinzip + Beispiel
  - Gehe den Text von links nach rechts durch
  - Prüfe an jeder Stelle ob das Muster passt, indem man es Buchstabe für Buchstabe mit dem Text dort vergleicht
  - Den jeweiligen Ausschnitt (der Größe m) aus dem Text nennen wir Fenster (engl. window)
  - Das implementieren wir jetzt zusammen !



Naiver Algorithmus 2/2

UNI FREIBURG

- Laufzeit
  - Sei wie gehabt n = Länge Text, m = Länge Pattern
  - Laufzeit im worst case + Beispiele:

    text = AAAAAAAA… pattern = AAA

    text = DUDADUDIDUDI… pattern = DUDA

    Man muss (oft) durch das ganze Pattern "durchgehen"
  - Laufzeit im best case + Beispiele:

    text = AACTAACCTAAGC pattern = XACAG

    Man merkt bei den ersten Buchstaben schon, dass das Pattern nicht passt

Knuth-Morris-Pratt Algorithmus 1/9

UNI FREIBURG

- Motivation für den Algorithmus
  - Wenn man die ersten k Zeichen des Musters mit den ersten k Zeichen eines Fensters im Text verglichen hat

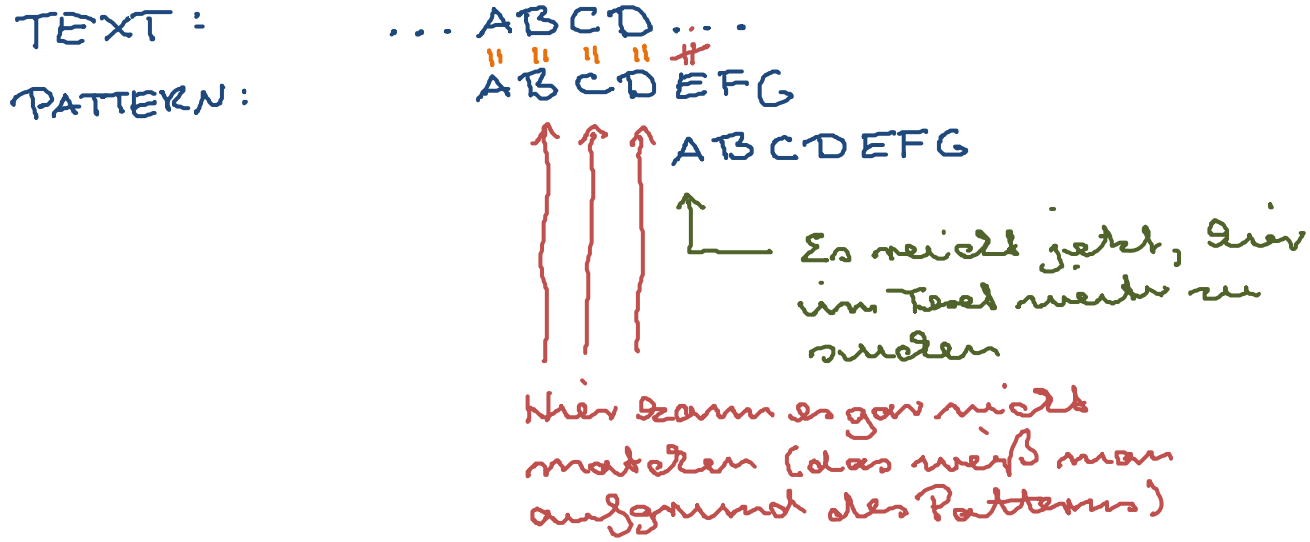
  ... und jetzt das Fenster im Text um eins weiter schiebt

  ... dann hat man k – 1 Zeichen dieses Fensters schon mal mit dem Muster verglichen
  - Man möchte gerne vermeiden, die nochmal anzuschauen
  - Wie das gehen könnte, sieht man am besten an ein paar Beispielen … nächste Folien

Knuth-Morris-Pratt Algorithmus 2/9

UNI FREIBURG

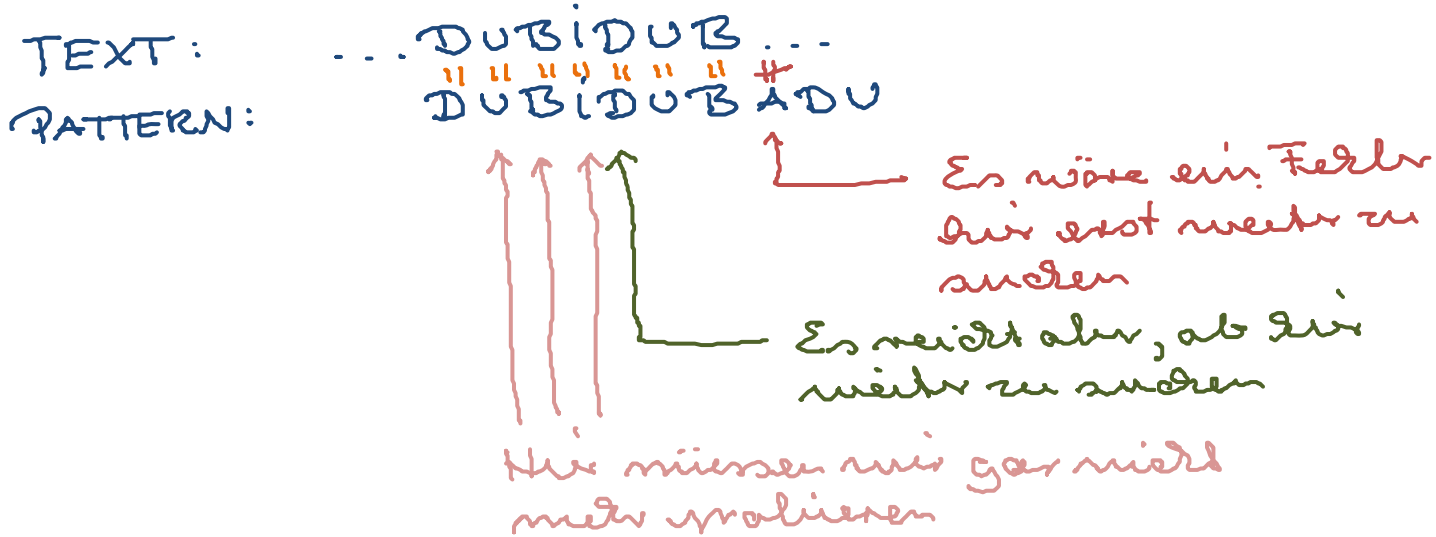
- Beispiel nicht-repetitives Muster
  - Im besten Fall kann man die Suche im Text da fortsetzen, wo der letzte "Mismatch" mit dem Muster war
  - Nehmen wir an, dass Muster ist **ABCDEFG**
  - Und nehmen wir an, an der aktuellen Textstelle passt es bis vor das **E** und dann nicht mehr **ABCDEFG**



Knuth-Morris-Pratt Algorithmus 3/9

UNI FREIBURG

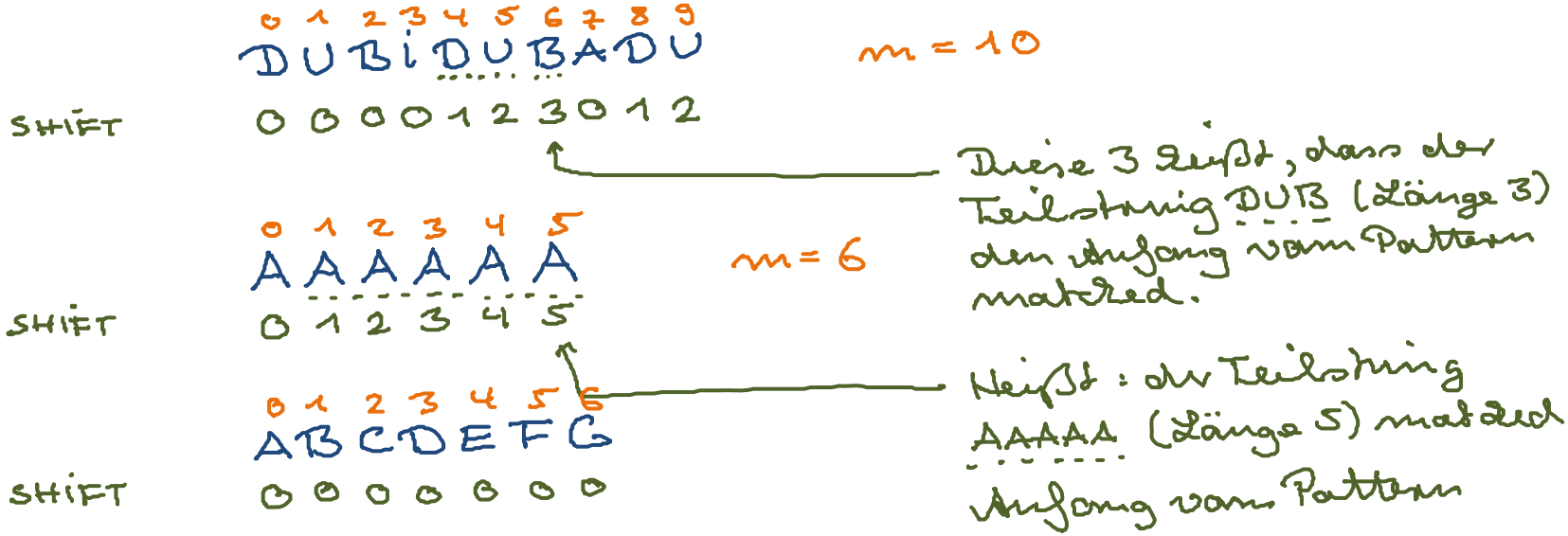
- Beispiel repetitives Muster
  - Es kann aber auch sein, dass es davor auch noch einen Treffer gibt
  - Nehmen wir an, dass Muster ist **DUBIDUBADU**
  - Und nehmen wir an, an der aktuellen Textstelle passt es bis vor das **A** und dann nicht mehr **DUBIDUBADU**



Knuth-Morris-Pratt Algorithmus 4/9

- Vorverarbeitung des Musters 1/3
  - Wir berechnen für jede Stelle des Musters vor, um wie viel links von der Stelle des letzten "Mismatches" man die Suche fortsetzen kann, ohne einen Treffer zu verpassen

    Dabei wollen wir so weit nach rechts gehen wie möglich
  - Erst mal ein paar Beispiele



Knuth-Morris-Pratt Algorithmus 5/9

UNI FREIBURG

- Vorverarbeitung des Musters 2/3
  - Genauer gesagt, berechnen wir für jedes $j \in \{0, \ldots, m - 1\}$

    $shift[j] = \max \{ k \leq j : P[j - k + 1 \,..\, j] = P[0 \,..\, k - 1] \}$

    In Worten: die Länge des längsten Teilstückes bis Stelle j (< alles bis j), die gleich dem Anfang des Musters ist
  - Man beachte, dass per Definition $shift[j] \leq j$

Knuth-Morris-Pratt Algorithmus 6/9

UNI FREIBURG

- Vorverarbeitung des Musters 3/3
  - Das Feld shift lässt sich einfach iterativ in Zeit O(m) von links nach rechts berechnen:

    Entweder shift[j] ist shift[j - 1] + 1

    Wenn das Teilstück, dass zu shift[j – 1] > 0 geführt hat auch noch bei pattern[j] passt

    Oder shift[j] ist 0 bzw. 1

    Je nachdem ob pattern[j] != pattern[0] oder nicht

Knuth-Morris-Pratt Algorithmus 7/9

UNI FREIBURG

- Beschreibung des Algorithmus
  - Vorberechnung des shift Feldes wie gerade erklärt
  - Genau wie beim naiven Algorithmus:

    Fenster der Größe m über den Text schieben

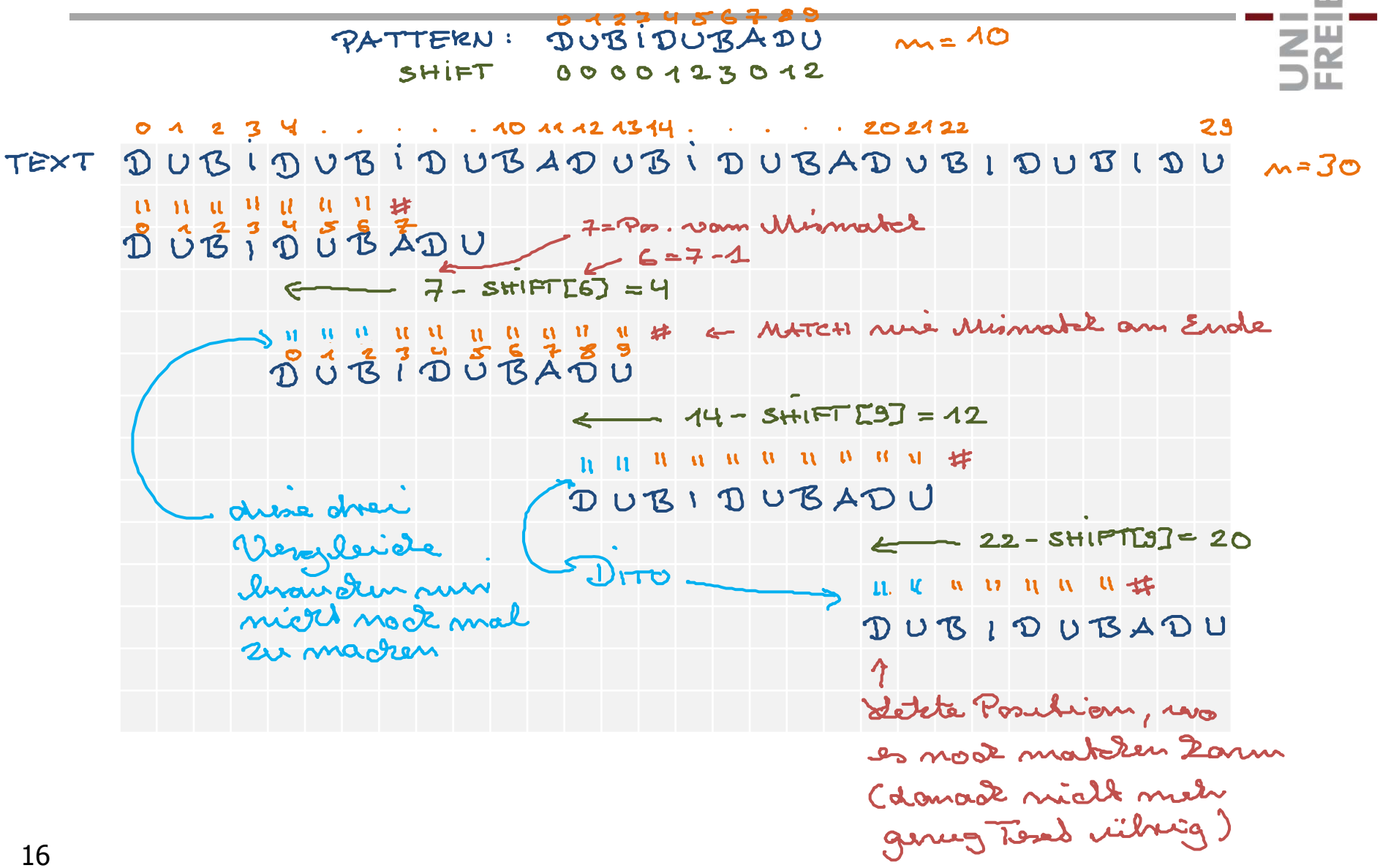
    An Stelle i prüfen ob das Muster passt
  - Einziger Unterschied zum naiven Algorithmus:

    Falls erster Mismatch an Stelle j in P, dann im Text weiter an Stelle $i + j - shift[j - 1]$ bzw. bei $i + 1$ falls $j = 0$

    Treffer dabei wie Mismatch bei $j = |P|$ behandeln

    Zum Vergleich: naiver Algo. macht immer bei $i + 1$ weiter

Knuth-Morris-Pratt Algorithmus 8a/9



Knuth-Morris-Pratt Algorithmus 8b/9

A A A X
" " " #
A A A A A A

Jetzt würde es sogar weiden,
als für weiter zu suchen

Die SHIFT werte sind nicht
optimal bei KMP
werden aber für Laufzeit O(n)

Knuth-Morris-Pratt Algorithmus 9/9

UNI FREIBURG

- Laufzeit
  - Die Laufzeit ist proportional zur Anzahl der Vergleiche

    eines Zeichens des Textes mit einem Zeichen des Patterns
  - Für jeden Vergleich gilt (siehe Bild auf Folie 16):

    Man schaut sich ein neues Zeichen im Text an

    Eins rechts von dem, dass man zuletzt verglichen hat

    Man schaut sich dasselbe Zeichen nochmal an, aber hat das Muster mindestens eins weiter "nach rechts geschoben"

    Die Verschiebung ist gerade $j - shift[j - 1] > 0$
  - Da man im Bild höchstens n mal "nach rechts" gehen kann, gibt es also höchstens $2n$ Vergleiche → Laufzeit $O(n)$

Literatur / Links

UNI FREIBURG

- String Matching
  - Mehlhorn/Sanders: gar nichts zu dem Thema!
- Wikipedia
  - http://de.wikipedia.org/wiki/Knuth-Morris-Pratt-Algorithmus
  - http://en.wikipedia.org/wiki/Knuth-Morris-Pratt_algorithm
- Originalarbeit
  - Donald Knuth und James Morris und Vaughan Pratt

    Fast Pattern Matching in Strings

    1977 SIAM Journal on Computing