Informatik II: Algorithmen und Datenstrukturen SS 2013

Vorlesung 12b, Mittwoch, 10. Juli 2013

(String Matching, Algorithmus von Knuth-Morris-Pratt)

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

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Weiter String Matching
  - Fortsetzung: Algorithmus von Rabin-Karp

    Laufzeitanalyse + Implementierungsdetails

    In der Implementierung gestern war noch ein **Veler** !
  - **Neu:** Algorithmus von Knuth-Morris-Pratt

    Beschreibung + Intuition + Beispiel + Laufzeitanalyse
  - Ü12, Aufgabe 2: Knuth-Morris-Pratt implementieren

Knuth-Morris-Pratt Algorithmus 1/9

UNI FREIBURG

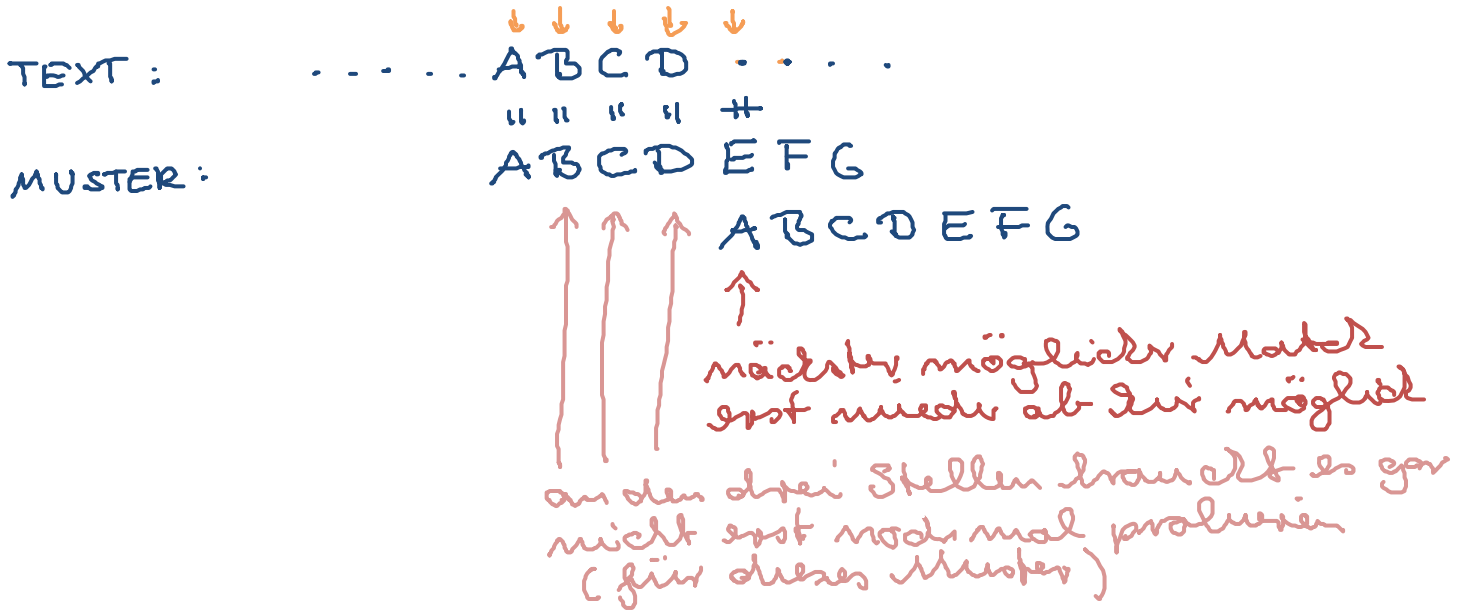
- Motivation für den Algorithmus
  - Wenn man die ersten k Zeichen des Musters mit den ersten k Zeichen eines Fensters im Text verglichen hat

  ... 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

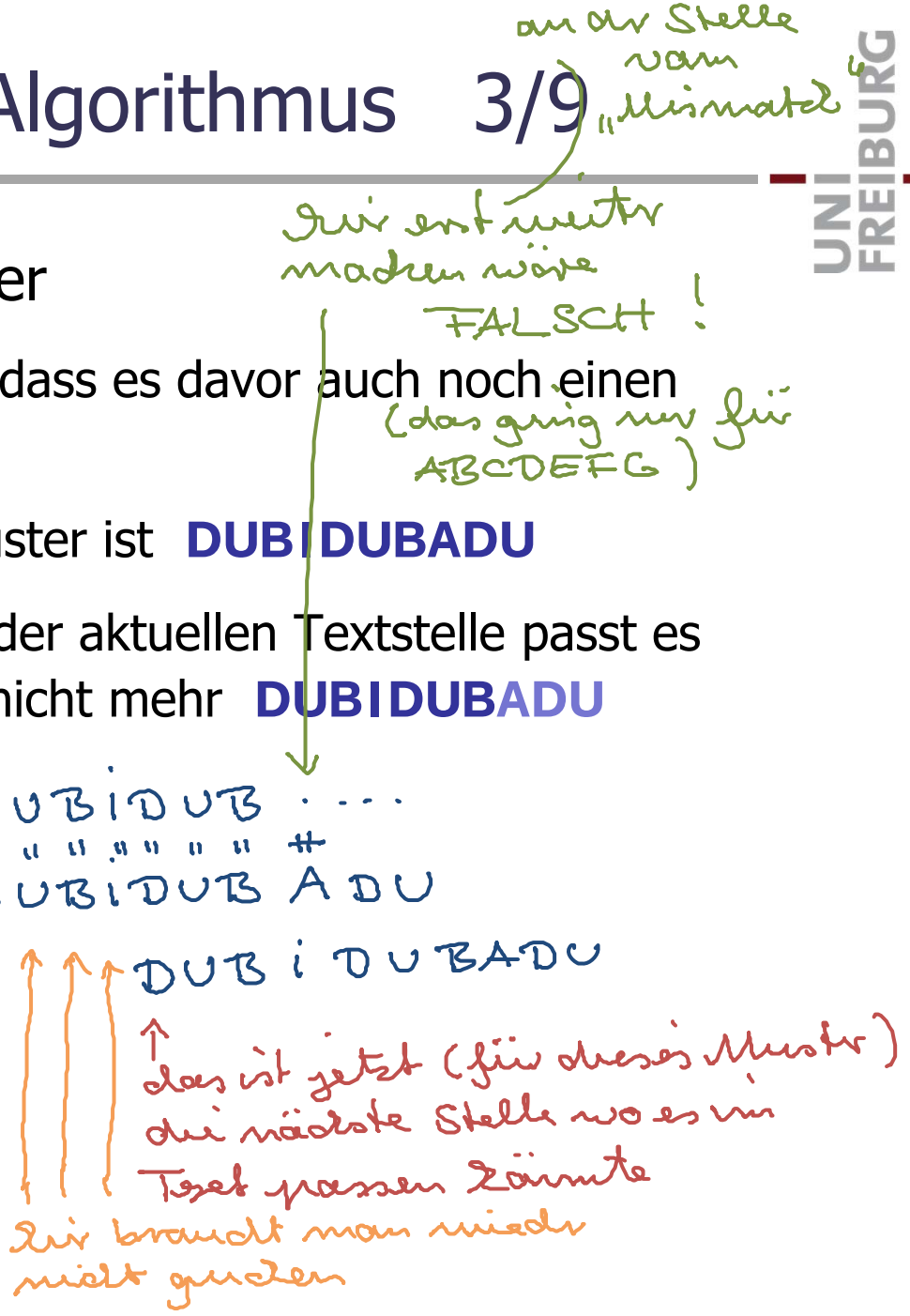
Knuth-Morris-Pratt Algorithmus 2/9

- 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

- 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**

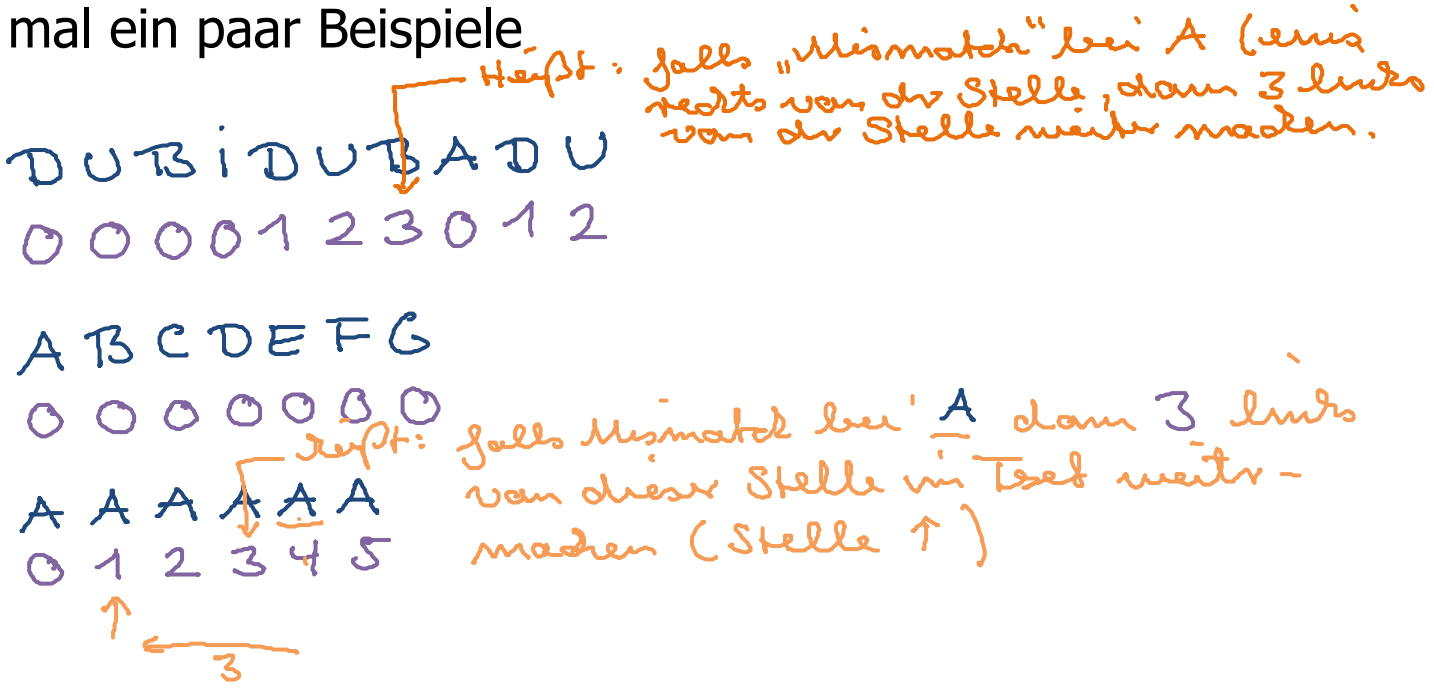


Knuth-Morris-Pratt Algorithmus 4/9

UNI FREIBURG

- 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 wenig nach links gehen wie möglich
  - Erst mal ein paar Beispiele



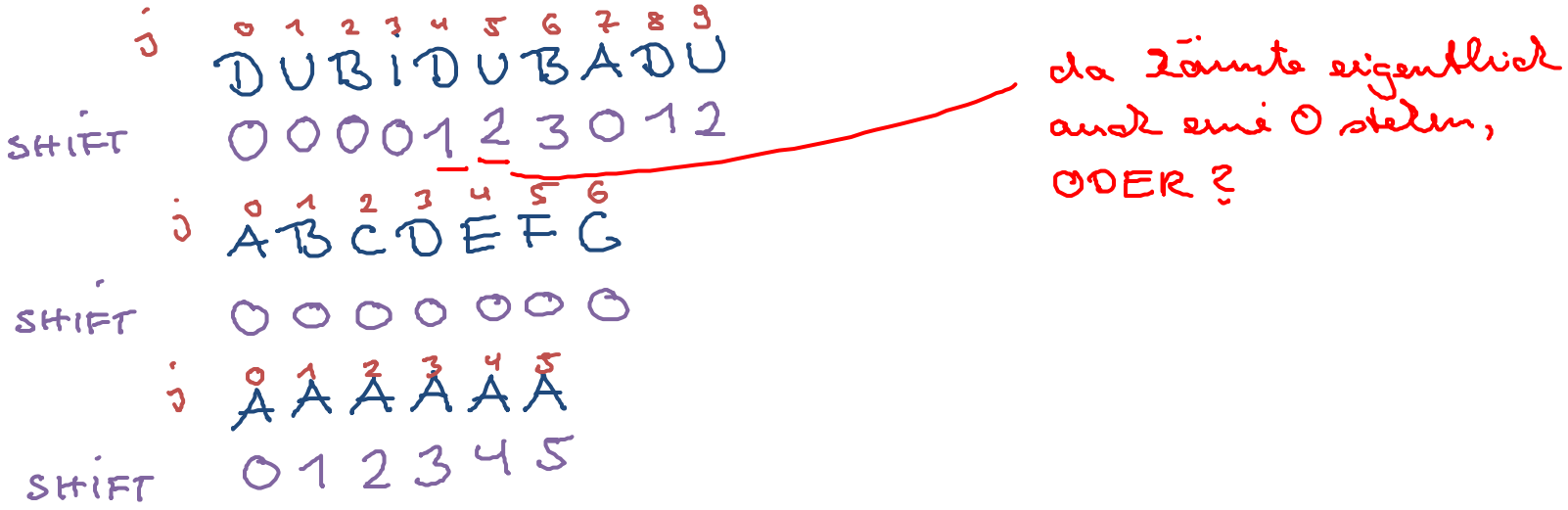
Knuth-Morris-Pratt Algorithmus 5/9

- 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 $shift[j] \leq j$

P = Muster



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
  - **Beobachtung:** shift[j + 1] ist entweder shift[j] + 1 oder 0, und das lässt sich anhand von P[j] entscheiden

    Der Rest ist Teil der Ü12, Aufgabe 2

    Schreiben Sie für die Methode precomputeShiftArray unbedingt einen separaten Unit Test !

    Sie können dazu einfach die Beispiele von der Folie vorher verwenden

Knuth-Morris-Pratt Algorithmus 7/9

UNI FREIBURG

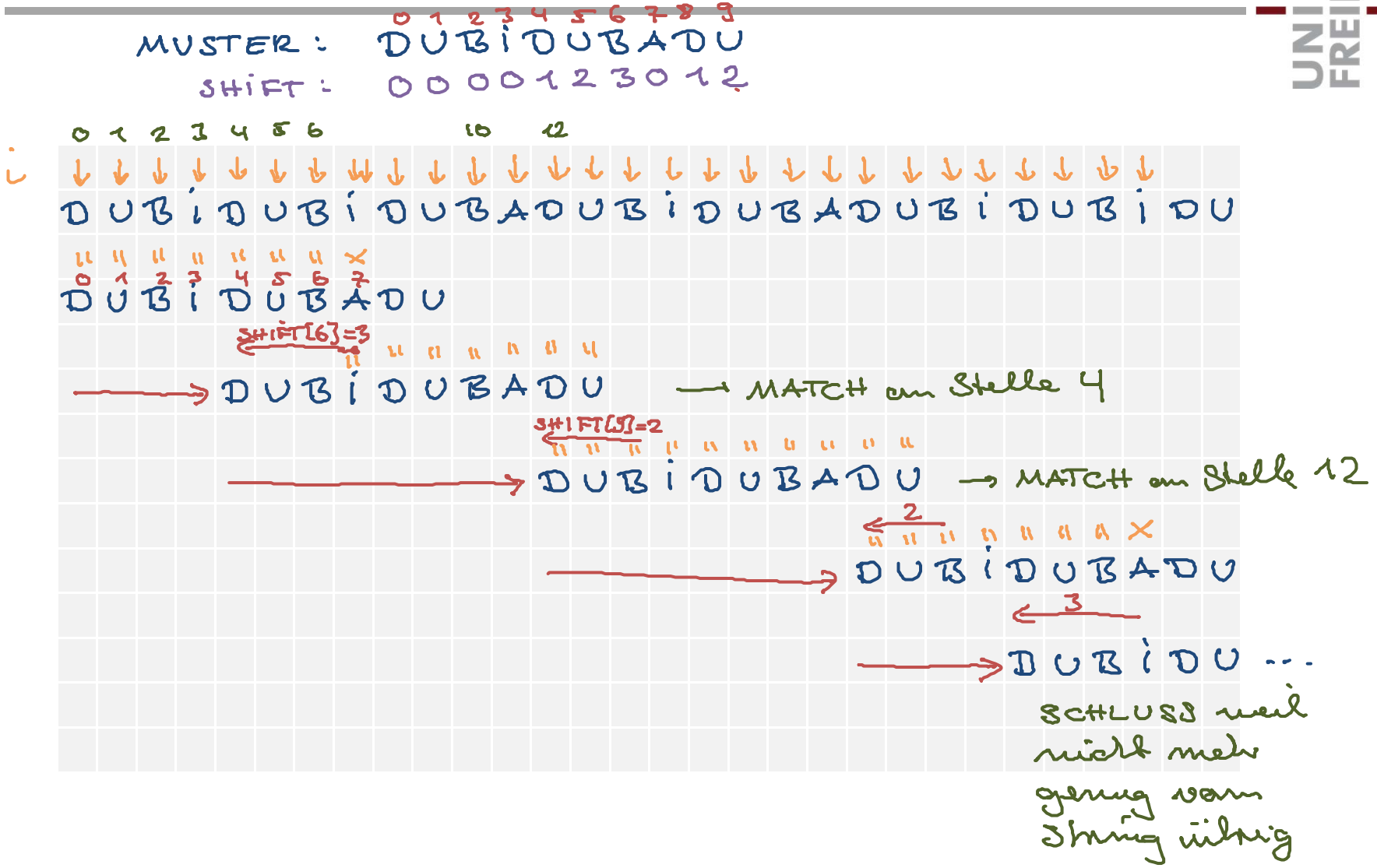
- Beschreibung des Algorithmus
  - Vorberechnung des shift Feldes wie gerade erklärt
  - Wie beim naiven Algorithmus ein Fenster der Größe m von links nach rechts über den Text schieben

    An Stelle i von links nach rechts prüfen ob das Muster passt, ebenfalls wie beim naiven Algorithmus

    Jetzt der Unterschied: falls erster Mismatch an Stelle j in P, dann an Stelle i – shift[j-1] im Text weiter machen (Match zählt wie Mismatch bei j = |P|) bzw. bei i + 1 falls j = 0

    Der naive Algorithmus macht immer an Stelle i + 1 weiter !
  - Dazu schauen wir uns jetzt ein vollständiges Beispiel an

Knuth-Morris-Pratt Algorithmus 8/9



Knuth-Morris-Pratt Algorithmus 9/9

UNI FREIBURG

- Laufzeit
  - Sei i der Index des aktuellen Zeichens im Text
  - In jeder Iteration erhöhen wir i oder es bleibt gleich
  - Wenn i gleich bleibt, verschieben wir das Muster im Bild (siehe Beispiel Folie 10) um mindestens eins nach rechts

    Die Verschiebung ist gerade $j - \text{shift}[j-1] > 0$
  - Da man im Bild höchstens n mal nach rechts gehen kann, gibt es also höchstens 2n Iterationen
  - In jeder Iteration gibt es nur konstant viele Operationen
  - Damit ist die Laufzeit O(n)

wohlgemerkt auch für

Text: AAAAA....

Muster: AAA

Literatur / Links

UNI FREIBURG

- 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