Informatik II: Algorithmen und Datenstrukturen SS 2015

Vorlesung 12b, Mittwoch, 15. Juli 2015

(String Matching, Teil 2)

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

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Inhalt
  - Knuth-Morris-Pratt Algorithmus … Fortsetzung von gestern
  - Karp-Rabin Algorithmus … Prinzip + Beispiele
  - Plagiatserkennung … praktische Anwendung von KR
  - Ü12, Aufgabe 1: KR implementieren
  - Ü12, Aufgabe 2: KR zur Plagiatserkennung einsetzen

Karp-Rabin Algorithmus 1/11

UNI FREIBURG

- Grundidee
  - Wir schieben wieder ein Fenster der Größe m über den Text und schauen an jeder Stelle, ob es zu dem Muster passt
  - Nehmen wir an die Buchstaben sind die Ziffern 0..9
  - Dann kann man das Muster als (große) ganze Zahl auffassen … und das Stück Text im aktuellen Fenster ebenso
  - Verschiebt man das Fenster um eins nach rechts, lässt sich die neue Zahl leicht aus der alten berechnen

m = 12 Text: 5 7 2 8 3 0 3 5 4 8 2 6

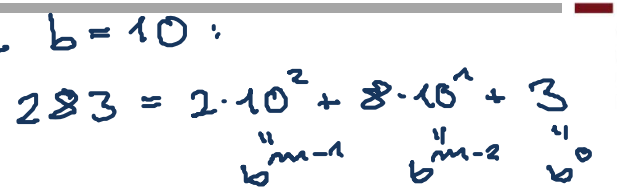
m = 3 Pattern: 2 8 3

× 283 ≠ 572 (= 72, = 720, = 728)

× 283 ≠ 728 (= 28, = 280, = 283)

✓ 283 = 283

Karp-Rabin Algorithmus 2/11



- Rechnen mit diesen "Zahlen"
  - Pro Fensterverschiebung braucht man nur konstant viele Rechen-Operationen auf diesen Zahlen
  - Wenn wir mit den Zahlen in konstanter Zeit operieren könnten, wäre die Laufzeit also O(n)

    Aber diese Zahlen können sehr groß werden

    Bei Basis b und einem Muster der Länge m bis zu b^m

    Dafür braucht man $\log_2 b^m = m \cdot \log_2 b$ Bits

    $\log_2 b = 8$

    Für b = 256 (ASCII) und m = 10 sind das schon 80 Bits …
    zu viel für z.B. ein int auf einem 64-Bit Rechner

Karp-Rabin Algorithmus 3/11

UNI FREIBURG

- Hashwerte
  - Statt Zahl x betrachten wir Hashwert $h(x) = x \bmod M$

    Die Hashwerte sind dann aus dem Bereich $\{0, \ldots, M - 1\}$
  - Bei einem Match sind sicher auch die Hashwerte gleich

    Aber gleiche Hashwerte bedeuten nicht unbedingt Match

    Wenn M groß ist, ist es allerdings unwahrscheinlich, dass ungleiche Zahlen auf denselben Wert abgebildet werden

    Das kennen wir ja schon vom Hashing (Kollisionen)
  - Wenn die Hashwerte gleich sind, überprüfen wir wie beim naiven Algorithmus Buchstabe für Buchstabe

Karp-Rabin Algorithmus 4/11

UNI FREIBURG

- Beispiel

  Text: 5 7 2 8 3 0 3 5 4 8 2 6

  Pattern: 2 8 3 $\quad h(283) = 3$

  Modulus: M = 5 , $\quad h(x) = x \bmod 5$

  In der Praxis benutzt man viel größere Werte für M

$h(572) = 2 \neq 3 = h(283)$ ⇒ SICHER kein Match

$h(728) = 3 = 3 = h(283)$ ⇒ Kandidat für ein Match, aber Nachprüfen der ursprünglichen Zahl (O(m) Zeit) → Kein Match

$h(283) = 3 = 3 = h(283)$ ⇒ Kandidat für Match und es ist auch ein (weiter O(m) Zeit)

Karp-Rabin Algorithmus 5/11

UNI FREIBURG

- Laufzeit mit Modulus $n = \#\text{Texts}$, $m = \#\text{Pattern}$
  - Im schlechtesten Fall: $O(n \cdot m)$

    Wenn Hashwert für Muster und Fenster immer gleich, obwohl das Muster gar nicht überall passt

    Bei guter Wahl von M unwahrscheinlich … spätere Folie
  - Im besten Fall: $O(n)$

    Das passiert, wenn der Modulus für das Muster und für das Textfenster nur dann gleich ist, wenn das Muster auch passt und „wenige" Matches

    Bei guter Wahl von M wahrscheinlich … spätere Folie

Karp-Rabin Algorithmus 6/11

UNI FREIBURG

- Wahl der Basis b und des Modulus M
  - Wir hätten gerne, dass wenn $x \neq y$, dann $h(x) = h(y)$ unwahrscheinlich, für $h(x) = x \bmod M$
  - Dazu sollte M möglichst groß sein und keinen Teiler mit b gemeinsam haben

    Z.B. falls b durch M teilbar, zählt nur die letzte "Stelle"
  - Wir wählen deshalb im Programm b = 257

    Kleinste Primzahl, die größer ist als jeder ASCII Code
  - Wir können dann M im Prinzip beliebig wählen

z.B. $b = 10$

$15127 \bmod 10 = 7$

$23417 \bmod 10 = 7$

$m = 4, b = 257$

$$\text{doof} = 100 \cdot 257^3 + 111 \cdot 257^2 + 111 \cdot 257^1 + 102 \cdot 1$$

$$b^{m-1} \quad b^{m-2} \quad b^1 \quad b^0$$

Ascii 100 111 111 102

Karp-Rabin Algorithmus 7/11

UNI FREIBURG

- Rechnen modulo M
  - Es gelten folgende Rechenregeln (für jedes $M \in \mathbf{N}$)

    $(a \cdot b) \bmod M = ((a \bmod M) \cdot (b \bmod M)) \bmod M$

    $(a + b) \bmod M = ((a \bmod M) + (b \bmod M)) \bmod M$

    Beweis: gute Aufgabe um Mathe zu üben !
  - Das heißt, wir können bei einem größeren Ausdruck das mod M auch schon auf Teilterme anwenden

    Das ist wichtig, damit die Zahlen nicht zu groß werden

Karp-Rabin Algorithmus 8/11

- Berechnung Hashwert in Java, C++, Python
  - In Java und C++ kann man statt einem expliziten mod M einfach den Überlauf beim Rechnen mit int nutzen

```
int hash_value = 0;
for (int j = 0; j < pattern.size(); j++) {
    hash_value = 257 * hash_value + pattern[j];
}
```

  - In Python können Zahlen beliebig groß werden, dort braucht man also ein explizites mod M

```
hash_value = 0
for c in pattern:
    hash_value = (257 * hash_value + c) % M
```

Karp-Rabin Algorithmus 9/11

$-38 \bmod 10$
$= -38 + 4 \cdot 10 = 2$

$715 \bmod 10$
$= 715 + (-71) \cdot 10 = 5$

- Modulo bei negativen Zahlen
  - Für eine beliebige ganze Zahl x ist mathematisch einfach

    $x \bmod m = \min \{ x + q \cdot m : q \in \mathbf{Z} \text{ und } x + q \cdot m \geq 0 \}$

    Damit ist immer $x \bmod m \in \{0, \ldots, m - 1\}$

    Zum Beispiel: $24 \bmod 10 = 4$ und $-4 \bmod 10 = 6$
  - In Java und C++ ist aber $x \,\%\, m = -(x \,\%\, m)$

    Zum Beispiel: $24 \,\%\, 10 = 4$ und $-4 \,\%\, 10 = -4$

    Wenn man es für Java oder C++ so macht, wie auf der Folie vorher, braucht man sich darum nicht zu kümmern
  - In Python ist $x \,\%\, m = x \bmod m$ … insbes. $-4 \,\%\, 10 = 6$

Karp-Rabin Algorithmus 10/11

UNI FREIBURG

- Erweiterung auf mehrere Patterns
  - Nehmen wir jetzt an, wir haben k Patterns und wollen alle Vorkommen **aller** dieser Patterns in einem Text finden
  - Eine typische Anwendung dafür ist die Plagiatserkennung

    Patterns = Sätze / Fragmente aus einer Originalarbeit

    Text = vermeintliches Plagiat
  - Wir könnte jetzt einfach einen unserer bisherigen Algorithmen k mal anwenden, für jedes Pattern einmal

    Die Laufzeit würde sich dann um dem Faktor k erhöhen, das dauert für große k (viele Patterns) sehr lange

    Für das Ü12, Aufgabe 2 ist k immerhin 288

Karp-Rabin Algorithmus 11/11

UNI FREIBURG

- Erweiterung von Karp-Rabin für k Patterns
  - Speichere die Patterns in einer Map, mit dem Hashwert als Schlüssel und dem Pattern als Wert
  - Für jedes Textstück (aus dem aktuellen Fenster), schauen wir dann, ob es Patterns mit demselben Hashwert gibt

    Das geht mit einer geeignete Map in O(1) Zeit
  - Falls ja, prüfen wir für jedes solche Pattern nach, ob es passt, wie beim naiven Algorithmus

    Für $M \gg k$ wird es selten vorkommen, dass der Hashwert übereinstimmt, aber das Pattern trotzdem nicht matched

    Die Laufzeit ist dann O(n + Gesamtlänge der Matches)

Literatur / Links

UNI FREIBURG

- Karp-Rabin Algorithmus
  - Wikipedia

    http://en.wikipedia.org/wiki/Rabin-Karp-Algorithmus
  - Originalarbeit von Rabin & Karp

    Richard Karp und Michael Rabin

    Efficient Randomized Pattern Matching

    1987 IBM Journal of Research and Development