Musterlösung
Übungszettel 4
Algorithmen & Datenstrukturen für Bioinformatiker

Adrian Haß

17. Februar 2003

1 Nochmal dfa und Regulärer Ausdruck

Zuerst sollte man sich einmal klar machen, dass es überhaupt möglich ist so einen dfa zu konstruieren.

Dies liegt daran, dass sich die Anzahl an 01- bzw. 10-Wörter höchstens um 1 unterscheiden kann. Dazu stelle man sich die Situation wie folgt vor: Wir haben zwei Inseln 0 und 1 und ich starte auf einer der beiden. Wie oft kann ich nun von 0 nach 1 (oder zurück) wechseln? Die Wechsel können sich nur um 1 unterscheiden, denn wenn ich in einem Schritt mal zur anderen gesprungen bin, dann muss ich offensichtlich erst wieder zurück, bevor ich wieder einen Sprung in dieser Richtung machen kann.

Diese Tatsache erlaubt es uns nun, einen dfa zu bauen, denn sonst können dfa's nicht zählen, weil sie nur ein "endliches" Gedächtnis haben. Dann überlegt man sich zusätzlich noch, dass die Differenz immer nur zugunsten des Worts verschoben sein kann, mit dessen Anfangsbuchstanden unser Wort angefangen hat. Das führt zu dem dfa von Abb.1.

2 Shift-And und `agrep`

(1) **'?' im Pattern:** Hier ist offensichtlich ein Buchstabe im Pattern, der alle des Textes matcht. Also können wir doch einfach in jedem der Vektoren $U(p)$, die angeben, wo Buchstabe p im Pattern vorkommt, an dieser Stelle eine 1 einfügen.

(2) **'?' im Text:** Hier gibt es offensichtlich ein Textstelle, die alle Buchstaben des Pattern matcht, also müssten wir hier eigentlich nur ein "shift" ausführen. Da wir aber nicht extra unsere Methode abändern wollen, führen wir einfach ein $U(*)$, das überall Einsen enthält, zu unseren Vektoren hinzu. Dies entsprich dann einem einzelnen "shift", da $x \wedge 1 = x$ gilt.

(3) **beides:** Einfach beide Methoden kombiniert.

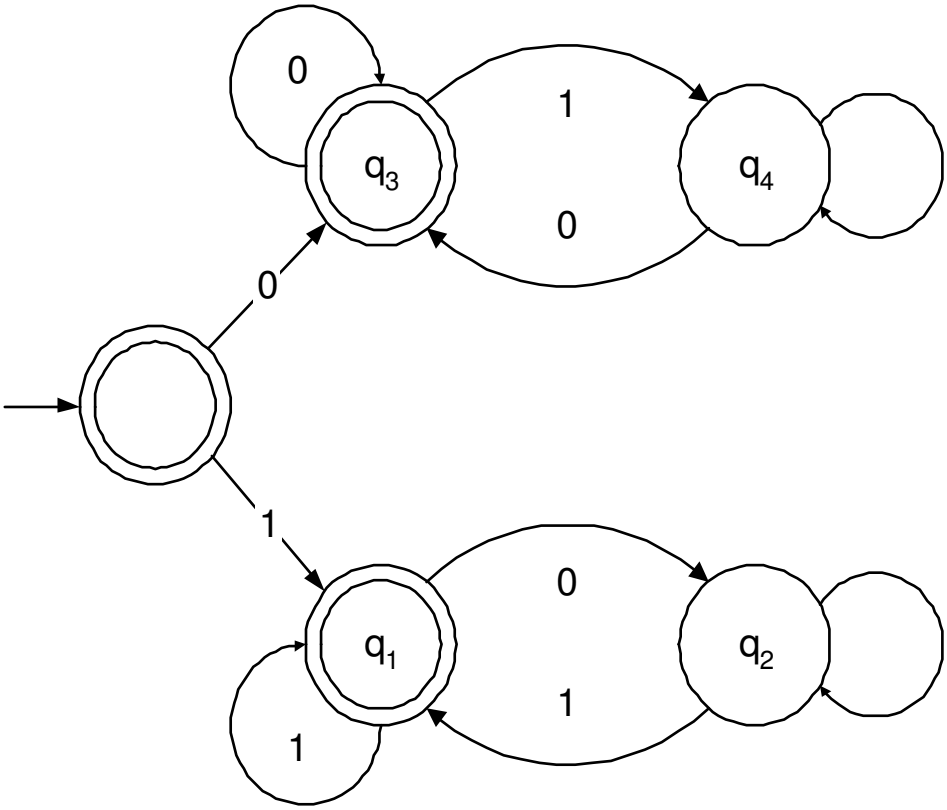

Abbildung 1: dfa für gleichviele 01/10-Wörter

3 Rechteckige Muster II

Wir haben hier also ein zweidimensionalen binären Text und ebensolche Bitmuster. Ausserdem haben die Muster alle gleiche Größe, denn sonst würde Karp-Rabin keinen Sinn machen.

Man überlegt sich nun, analog zum eindimensionalen Fall, auch Rechenregeln um einen Fingerprint vom Text in der Größe der Muster beim Verschieben nicht immer neu zu berechnen. Diesen wird man dann geeignet über den Text schieben und mit den Prints der Pattern vergleichen.

Die Prints werden einfach Zeilenweise berechnet und dann wird noch die i-te Zeile mit $2^{l \cdot i}$ multipliziert (l ist die Breite des Fensters). Also können wir Verschieben nach oben durch abziehen der vormals untersten Zeile, adddieren der neuen obersten und multiplizieren aller anderen Zeilen mit 2^l bewerkstelligen. Für die Rechtsverschiebung ist es ähnlich, nur ist das gleiche mit der Spalte zu erledigen. Dafür wird jeder erste Buchstabe einer Zeile mit der entprechenden Potenz abgezogen und die neuen addiert.

Warum kann man nicht nach unten/links verschieben? Wir haben zu bedenken, dass die Operationen nicht explizit ausgeführt werden, sondern modulo der gewählten Primzahl p. Beim Multiplizieren haben wir kein Problem, denn wenn p eine Zahl teilt, dann teile p auch alle Vielfache. Beim dividieren, was wir bei links/unten tun müssen, geht das nicht!

Beim Verschieben hat man nun auf den ersten Blick die Wahl, wie man das Fenster über den Text schiebt. Beim zweiten stellt man jedoch fest, dass einem das HORNER-Schema nur beim hochschieben viel hilft. Also wählen wir die Variante, wo wir möglichst wenig nach rechts schieben und viel nach oben: Start ist links unten, 1. Spalte (untersten Fingerprint merken!) nach oben, unerste Zeile eins nach rechts, 2. Spalte nach oben...

Verschiedene Lösungen basierten auf einem Zeilenweisen Karp-Rabin und dann einem Aho-Corasick wie auf dem letzten Zettel. Diese Variante ist deshalb schlecht, weil man dann eine Primzahl zeilenweise bestimmt und für die Zeile dann eine Fehler W'keit hat. Diese addieren sich jedoch für das rechteckige Muster. In der beschriebenen Variante ist das nicht so, da man die Primzahl für das ganze Rechteck wählt.

4 Matching mit Suffixbaum

Klar ist, dass sich in $O(m)$ festellen lässt, ob P vorkommt (Vergleiche von oben). Bleibt zu zeigen, dass sich die k vorkommen in Linearzeit ermitteln lassen.

Da den Vorkommen jeweils Blätter des Teilbaumes unter der aktuellen Position entsprechen, müssen diese noch mit In-/Pre- oder Postorder durchlaufen werden. Diese ist aber nur linear in der Anzahl der Knoten (innnere und Blätter) des Baums. Jedoch ist die Anzahl der inneren Knoten n eines Baumes immer kleiner als die Anzahl der Blätter k, also die Knotenzahl $< 2k$. Somit läuft die anschliessende Traversieung in $O(k)$, also ist demnach die Gesamtlaufzeit $O(m + k)$ wie gewünscht.

Ein Beispiel finde jeder selbst zur weiteren Übung.