Universität Stuttgart

Übungsblatt 9

Datenstrukturen und Algorithmen (SS 2022)

Abgabe: Montag, 27.06.2022, 15:30 Uhr — Besprechung: ab Montag, 04.07.2022

Abgabevorschriften: Die Aufgaben auf diesem Blatt sind unter Einhaltung der Abgabevorschriften[1] zu lösen und abzugeben.

Lernziele: Nach dem Tutorium zu diesem Blatt sollten Sie folgende Lernziele erreicht haben. Wenn nicht, zögern Sie nicht, Ihre:n Tutor:in anzusprechen um die Lücken zu füllen, Unklarheiten zu klären oder Fragen zu beantworten.

- Sie kennen die Vor- und Nachteile vom Brute-Force-, Knuth-Morris-Pratt- und Boyer-Moore-Algorithmus. Sie können diese Algorithmen anwenden, ggfs die zugehörigen Tabellen aufstellen und in Java implementieren.
- Sie können reguläre Ausdrücke aufstellen und auswerten.
- Sie kennen den Unterschied zwischen nichtdeterministischen endlichen Automaten und deterministischen endlichen Automaten, können sie formal aufschreiben, ineinander umwandeln, verwenden und in Java implementieren.
- Sie können die Levenshtein-Distanz erklären, anwenden und in Java implementieren.

Punkte: Dieses Übungsblatt beinhaltet 5 Aufgaben mit einer Gesamtzahl von 30 Punkten. Zum Bestehen werden also 15 Punkte benötigt.

Aufgabe 1 Knuth-Morris-Pratt *[Punkte: 4]*

In dieser Aufgabe werden Sie die Vorteile des Knuth-Morris-Pratt-Algorithmus, den Sie in der Vorlesung zum Thema String-Matching kennen gelernt haben, benennen und ihn auf Beispielen ausführen.

(a) *(1 Punkt)* Was macht sich der *Knuth-Morris-Pratt-Algorithmus* verglichen mit dem *Brute-Force-Algorithmus* beim String-Matching zu Nutze? Erläutern Sie.

(b) *(1 Punkt)* Stellen Sie die *Präfixtabelle* für das Pattern p_1 = "`aabaabbabcababd`" auf.

(c) *(2 Punkte)* Führen Sie den *Knuth-Morris-Pratt-Algorithmus* durch, um das Pattern p_2 = "`infoinfoinf`" in dem Text t_2 = "`inffinfoinfcinfiinfoininifinfoinfoinf`" zu finden. Hierfür sei die zugehörige *Präfixtabelle* gegeben. Stellen Sie dar, wie Sie vorgegangen sind und dokumentieren Sie Ihr Vorgehen. Orientieren Sie sich dabei an der Darstellungsweise aus der Vorlesung.

Position k in p_2	0	1	2	3	4	5	6	7	8	9	10
Pattern $p_1[k]$	i	n	f	o	i	n	f	o	i	n	f
prefix[k]	0	0	0	0	1	2	3	4	5	6	7

Aufgabe 2 Boyer-Moore *[Punkte: 6]*

In dieser Aufgabe werden Sie die Vorteile des Boyer-Moore-Algorithmus, den Sie in der Vorlesung zum Thema String-Matching kennengelernt haben, benennen und ihn auf Beispielen stückweise ausführen.

(a) *(1 Punkt)* Was macht sich der *Boyer-Moore-Algorithmus* verglichen mit dem *Knuth-Morris-Pratt-Algorithmus* zu Nutze, um effizienter zu sein? Erläutern Sie.

(b) *(3 Punkte)* Geben Sie für das Suchmuster p_3 = "`ininffoinf`" und den Text t_3 = "`naffincffinfo`" die entsprechende *last-*, *suffix-* und *shift-Tabelle* an. Geben Sie die last-Tabelle für alle im Beispiel verwendeten Zeichen an.

[1] `https://ilias3.uni-stuttgart.de/goto_Uni_Stuttgart_file_2904210.html`

(c) *(2 Punkte)* Führen Sie den *Boyer-Moore-Algorithmus* aus, um das Pattern p_4 = "`xxyzxyz`" in dem Text t_4 = "`wxxyzxyxxvxyzyyyzxzxyzaxxyzxyzxxxyzxyz`" zu finden. Hierfür sei die zugehörige *last-* sowie *shift-Tabelle* gegeben. Stellen Sie dar, wie Sie vorgegangen sind und dokumentieren Sie Ihr Vorgehen. Geben sie bei jedem Sprung an, wie groß die Beiträge aus den Heuristiken *Good Suffix/Match* und *Bad Character/Occurence* (in den Folien die Berechnung der entsprechenden shift und last Werte) des Algorithmus sind.

c	a	v	w	x	y	z
last[c]	-1	-1	-1	4	5	6

i	0	1	2	3	4	5	6
$p_4[i]$	x	x	y	z	x	y	z
shift[i]	7	7	7	3	7	7	1

Aufgabe 3 Endliche Automaten *[Punkte: 8]*

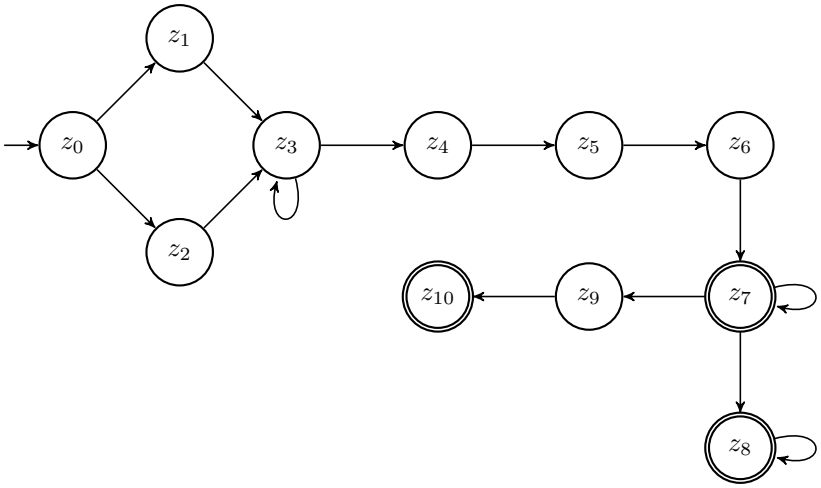
In dieser Aufgabe werden Sie mit (nicht)deterministischen endlichen Automaten arbeiten und reguläre Ausdrücke aufstellen.

Hinweise: Sehen Sie sich für die Aufstellung der regulären Ausdrücke die Notation auf den Vorlesungsfolien genau an. Denken Sie außerdem daran, die Zeichen für Zeilenbeginn und -ende sowie nötige Klammern einzufügen.

(a) *(2 Punkte)* Zu einem *nichtdeterministischen endlichen Automaten* M_2 sei der folgende *regulärer Ausdruck* bekannt, der alle von M_2 erkannten Wörter beschreibt:

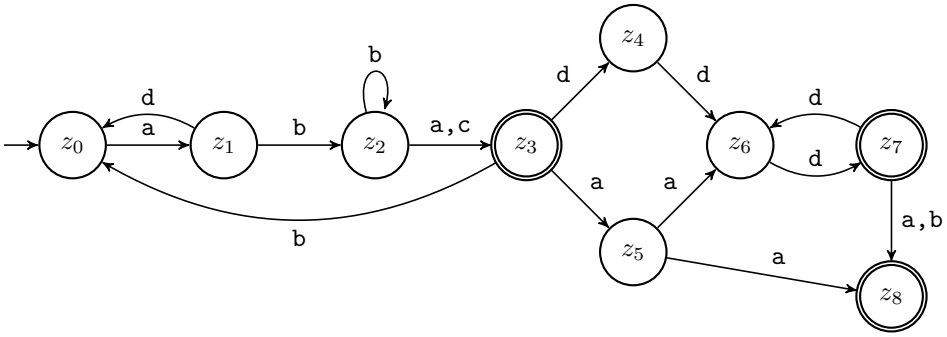
`^(bc|cb)(a|b)□[dc]ab(c□)(ba|d□)?$`

In diesem Ausdruck selbst sind leider ein paar Zeichen abhandengekommen (gekennzeichnet durch □). Von dem M_2 sei zudem die folgende Struktur bekannt:



Vervollständigen Sie M_2, indem Sie alle Übergänge beschriften. Sie sollen dabei M_2 weder Übergänge noch Zustände hinzufügen oder sonstige Modifikationen tätigen, sondern lediglich die Übergänge richtig beschriften. Geben Sie auch den vervollständigten *regulären Ausdruck* an (also ohne □).

(b) Gegeben Sei der folgende *endliche Automat* M_1:

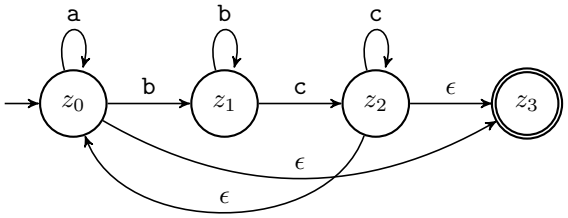


i. *(1 Punkt)* Handelt es sich bei M_1 um einen *deterministischen endlichen Automat* oder um einen *nichtdeterministischen endlichen Automat*? Begründen Sie Ihre Antwort.

ii. *(1 Punkt)* Geben Sie M_1 formal als 5-Tupel $M_1 = (S, \Sigma, z_0, F, move)$ an. Definieren Sie dafür die Mengen bzw. Funktionen im 5-Tupel.

iii. *(1 Punkt)* Geben Sie den zugehörigen *regulären Ausdruck* an, der alle von M_1 erkannten Worte beschreibt.

(c) *(1 Punkt)* Gegeben sei der *nichtdeterministische Automat* M_3. Konstruieren Sie einen äquivalenten *deterministischen Automaten* M_4 mit maximal 5 Zuständen und geben Sie den zugehörigen *regulären Ausdruck* an.



(d) Geben Sie folgende reguläre Ausrücke an:

i. *(0.5 Punkte)* Alle Wörter der Länge 5, deren zweites Zeichen ein a oder b ist und mit ab enden.

ii. *(0.5 Punkte)* Alle Telefonnummern aus Deutschland (Vorwahl: 0049), Österreich (Vorwahl: 0043), Italien (Vorwahl: 0039) und Australien (Vorwahl: 0061) mit einer Gesamtlänge von mindestens 10 und maximal 15 Zeichen. Das Alphabet sei hierbei gegeben durch $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

iii. *(0.5 Punkte)* Alle Zeichenketten, die folgendes erfüllen:
- Die Zeichenkette beginnt mit `aa` oder `bb`
- Es folgt ein c.
- Es fogt optional ein weiteres c.
- Es folgt entweder eine gerade Anzahl größer 2 von a's oder eine ungerade Anzahl größer 1 von b's.
- Es fogt eine beliebige Anzahl an a's größer 0.
- Die Zeichenkette endet optional mit einem $bbbb$.

iv. *(0.5 Punkte)* Alle Wörter über dem Alphabet $\Sigma = \{a, b\}$, wobei die Anzahl des Zeichens a durch ungerade sein muss.

Aufgabe 4 Impl Nichtdeterministische endliche Automaten implementieren *[Punkte: 7]*

In dieser Aufgabe sollen Sie sich mit der Implementierung von *nichtdeterministischen endlichen Automaten* vertraut machen. Hierfür sind im Eclipse-Projekt folgende Codefragmente gegeben:

- Unvollständige Klasse `NFA`

(a) *(3 Punkte)* Implementieren Sie die Methode `concat`. Diese bekommt zwei *nichtdeterministische endliche Automaten* übergeben und soll diese zu einem *nichtdeterministischen endlichen Automaten* konkatinieren.

(b) *(2 Punkte)* Implementieren Sie die Methode `disjunction`. Diese bekommt zwei *nichtdeterministische endliche Automaten* übergeben und soll diese durch Disjunktion zu einem andere *nichtdeterministischen endlichen Automaten* umwandeln.

(c) *(2 Punkte)* Implementieren Sie die Methode `repitition`. Diese bekommt einen *nichtdeterministische endliche Automaten* übergeben und soll die kleensche Hülle als *nichtdeterministische endliche Automaten* zurückgeben.

Hinweise:

- Lesen Sie den vorgegebenen Code sorgfältig.
- Modifizieren Sie ausschließlich die zu implementierenden Methoden der Klasse `NFA`. Alle anderen Klassen und Interfaces dürfen **nicht** verändert werden.

- Verwenden Sie für alle zu implementierenden Methoden die bereits implementierte Methode `addTransition`.
- Überlegen Sie sich beliebige *nichtdeterministische endliche Automaten* und führen Sie Ihre implementierten Methoden aus, um Ihren Code auf Korrektheit zu prüfen.

Aufgabe 5 Levenshtein Distanz *[Punkte: 5]*

In dieser Aufgabe werden Sie Verwandte der Levenshtein-Distanz kennenlernen und anwenden.

(a) *(2 Punkte)* Wenden Sie eine leicht abgeänderte Form des Algorithmus zur Berechnung der *Levenshtein-Distanz* auf die Eingabe "Fachschaft" und "Fachrat" an. Hierbei sollen Operationen nicht gleich teuer sein. Es gelte:

$$D_{i,j} = min \begin{cases} D_{i-1,j-1} + 0 & \text{falls gleiches Zeichen} \\ D_{i-1,j-1} + 2 & \text{falls Zeichen ersetzen} \\ D_{i,j-1} + 1 & \text{falls Zeichen einfügen} \\ D_{i-1,j} + 1 & \text{falls Zeichen löschen} \end{cases}$$

Füllen Sie die gegebene Tabelle aus. Geben Sie außerdem die Levenshtein-Distanz der gegebenen Eingabe an und welche Operationen auf "Fachschaft" angewendet werden müssten, um "Fachrat" zu erhalten.

	ϵ	F	a	c	h	s	c	h	a	f	t
ϵ											
F											
a											
c											
h											
r											
a											
t											

(b) Die *Damerau-Levenshtein-Distanz* ist der *Levenshtein-Distanz* sehr ähnlich. Allerdings werden bei der *Damerau-Levenshtein-Distanz* auch Transpositionen berücksichtigt. Unter einer Transposition wird hierbei eine Permutation von zwei Zeichen verstanden. Beispielsweise befindet sich in den Zeichenketten "Daten" und "Dtaen" eine Transposition am zweiten und dritten Zeichen. Für die *Damerau-Levenshtein-Distanz* ergibt sich somit folgendes:

$$D_{i,j} = min \begin{cases} D_{i-1,j-1} + 0 & \text{falls gleiches Zeichen} \\ D_{i-1,j-1} + 1 & \text{falls Zeichen ersetzen} \\ D_{i,j-1} + 1 & \text{falls Zeichen einfügen} \\ D_{i-1,j} + 1 & \text{falls Zeichen löschen} \\ D_{i-2,j-2} + 1 & \text{falls Transposition mit vorhergehendem. Zeichen} \end{cases}$$

.

Hinweis: Die Kosten für das Ersetzen eines Zeichens sind in dieser Teilaufgabe wieder 1.

i. *(1 Punkt)* Nennen Sie einen Vorteil der *Damerau-Levenshtein-Distanz* gegenüber der *Levenshtein-Distanz* und begründen Sie diesen. Eine Antwort ohne Begründung wird mit null Punkten bewertet.

ii. *(2 Punkte)* Wenden Sie den Algorithmus zur Berechnung der *Damerau-Levenshtein-Distanz* auf die Eingabe "Comtimnett" und "Commitment" an. Füllen Sie die gegebene Tabelle aus. Geben Sie außerdem die Damerau-Levenshtein-Distanz der gegebenen Eingabe an und welche Operationen auf "Comtimnett" angewendet werden müssten, um "Commitment" zu erhalten.

	ϵ	C	o	m	m	i	t	m	e	n	t
ϵ											
C											
o											
m											
t											
i											
m											
n											
e											
t											
t											