Universität Bonn, Institut für Informatik I
MUSTERLÖSUNG 2. Klausur Informatik IV SS 2007
26.9.2007, 90 Minuten

NAME	VORNAME	Matrikelnummer
Lösung	Muster	

Zutreffendes bitte ankreuzen:

☐ Ich stimme zu, dass mein Klausurergebnis zusammen mit meiner Matrikelnummer im Internet veröffentlicht wird.

☐ Ich stimme der Veröffentlichung meines Klausurergebnisses im Internet **nicht** zu.

☐ Neue DPO ☐ Alte DPO ☐ Nebenfach

Übungsgruppe 1	Mo, 15:00–17.00 (Ivan)	
Übungsgruppe 2	Mo, 17:00–19.00 (Anette)	
Übungsgruppe 3	Di, 15:00–17.00 (Andreas T.)	
Übungsgruppe 4	Mi, 09:00–11.00 (Stefan)	
Übungsgruppe 5	Do, 09:00–11.00 (Georg)	
Übungsgruppe 6	Do, 11:00–13.00 (Andreas L.)	
Übungsgruppe 7	Do, 13:00–15.00 (Daniel)	
Übungsgruppe 8	Fr, 12:00–14.00 (Uwe)	
Übungsgruppe 9	Fr, 14:00–16.00 (Rebecca)	

Nachstehende Tabelle bitte NICHT ausfüllen!

1	2	3	4	5	6	7	8	Summe	Note
10	10	10	10	10	10	10	10	80	

Aufgabe 1 [Multiple Choice, 10 Punkte]

(pro Teilaufgabe 2 Punkte) Kreuzen Sie alle zutreffenden Aussagen an! Es können zu einem Unterpunkt mehrere Aussagen, aber auch keine einzige Aussage richtig sein.

a) Welche der folgenden Aussagen ist äquivalent dazu, dass der ungerichtete Graph $T = (V, E)$ ein Baum ist?

- ☐ In T taucht keine Kante doppelt auf.
- × T ist zusammenhängend und kreisfrei.
- ☐ Jedes Paar von Knoten aus V ist in T durch mindestens einen Pfad verbunden.
- × $|E| = |V| - 1$ und T ist zusammenhängend.

b) Jeder Sweep-Algorithmus benutzt ...

- ☐ eine Sweepline-Randomisierung.
- × eine Sweep-Status-Struktur.
- ☐ eine Rekursion.
- × eine Ereignisstruktur.

c) Welche der folgenden Aussagen über randomisierte Algorithmen treffen zu?

- × Ein Las-Vegas-Algorithmus liefert nie ein falsches Ergebnis.
- ☐ Ein Monte-Carlo-Algorithmus liefert nie ein falsches Ergebnis.
- ☐ Ein Monte-Carlo-Algorithmus hat nur im Erwartungswert eine günstige Laufzeit, im Worst-Case kann sie schlecht sein.

d) Eine Sprache $L \subseteq \Sigma^*$ für ein endliches Alphabet Σ ist genau dann regulär, wenn ...

- × es einen nichtdeterministischen endlichen Automaten M gibt mit $L(M) = L$.
- × es einen deterministischen endlichen Automaten M gibt mit $L(M) = L$.
- ☐ es eine kontextfreie Grammatik G gibt mit $L(G) = L$.
- × es einen regulären Ausdruck α über Σ gibt mit $L(\alpha) = L$.

e) Für jede natürliche Zahl $n \in \mathbb{N}$ gilt:

- ☐ n ist Primzahl $\Leftrightarrow (\forall a : 1 \leq a \leq n-1 \Rightarrow a^{n-1} \bmod n = 1)$.
- × n ist Primzahl $\Rightarrow (\forall a : 1 \leq a \leq n-1 \Rightarrow a^{n-1} \bmod n = 1)$.
- ☐ n ist Primzahl $\Leftarrow (\forall a : 1 \leq a \leq n-1 \Rightarrow a^{n-1} \bmod n = 1)$.

Aufgabe 2 [Sortieralgorithmen, 10 Punkte]

a) Sortieren Sie die folgenden Zahlen mit dem Quicksort-Verfahren, das in der Vorlesung vorgestellt wurde:

15 42 4 16 8 23

Für eine Folge von Zahlen werde stets das **erste** Element dieser Folge als Pivotelement ausgewählt. Geben Sie in jedem Schritt die Position der Indizes I und T sowie das Pivotelement an, kennzeichnen Sie außerdem, welche Teilfolge gerade bearbeitet wird.

b) Beschreiben Sie (stichpunktartig)

(i) das Mergesort-Verfahren.

(ii) die wesentlichen Unterschiede zwischen randomisierten Quicksort und Mergesort.

(iii) die wesentlichen Gemeinsamkeiten zwischen randomisierten Quicksort und Mergesort.

Kategorien zu (ii) und (iii) sind Arbeitsweise, Paradigma, Laufzeit, Speicherbedarf.

Antwort:

a)

[15]	42	4	16	8	23	∞
	$\uparrow I$			$\uparrow T$		
[15]	8	4	16	42	23	∞
		$\uparrow T$	$\uparrow I$			
[4]	8	15	16	42	23	∞
[4]	8					
$\uparrow T$	$\uparrow I$					
			[16]	42	23	
			$\uparrow T$	$\uparrow I$		
				[42]	23	∞
					$\uparrow T$	$\uparrow I$
				23	42	∞
4	8	15	16	23	42	

b)

Mergesort:
- Teile die Zahlenfolge in 2 Teilfolgen mit je (etwa) $\frac{n}{2}$ Elementen.
- Sortiere beide Folgen rekursiv.
- Mische die beiden Folgen zu einer Folge.

Unterschiede:

Quicksort	Mergesort
• Arbeit im Divide-Schritt	• Arbeit im Merge-Schritt
• Worst-Case Laufzeit $O(n^2)$	
• Zus. Speicher $O(1)$	• Zus. Speicher $\Omega(n)$
• Randomisiert	• Deterministisch

Gemeinsamkeiten:
- Beides Divide and Conquer Algorithmen
- (erwartete) Laufzeit $O(n \log n)$

Aufgabe 3 [Automaten, 10 Punkte]

a) Gegeben sei ein NEA $M_1 = (Q_1, \Sigma_1, \delta_1, q_{0,1}, F_1)$. Geben Sie **formal** einen dazu äquivalenten DEA $M_1 = (Q_2, \Sigma_2, \delta_2, q_{0,2}, F_2)$ an.

b) Die Sprache $L \subset \{a, b\}^*$ sei wie folgt definiert: $L = \{abwab | w \in \{a, b\}^*\}$.

Geben Sie einen **minimalen** DEA an, der diese Sprache akzeptiert. D.h. geben Sie die Zustandsübergangsfunktion δ als Zustandsdiagramm an und geben Sie das Tupel an, das den Automaten beschreibt.

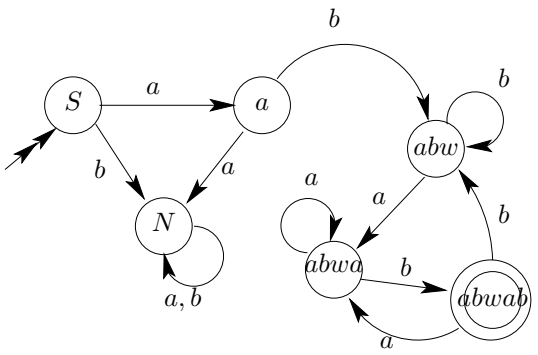
Antwort:

a)

$$
\begin{aligned}
&Q_2 := \mathcal{P}(Q_1) \\
&\Sigma_2 := \Sigma_1 \\
&q_{0,2} := R_\epsilon(q_{0,1}) \\
&F_2 := \{A \subseteq Q_1 | A \cap F_1 \neq \emptyset\} \\
&\delta_2(A, a) := R_\epsilon\left(\delta_1(A, a)\right) \text{ mit } A \subseteq Q_1 \text{ und } a \in \Sigma_1
\end{aligned}
$$

b) $(Q, \Sigma, \delta, q_0, F)$ mit

$$
\begin{aligned}
&Q = \{S, N, a, abw, abwa, abwab\} \\
&\Sigma = \{a, b\} \\
&F = \{abwab\} \\
&q_0 = S
\end{aligned}
$$



Aufgabe 4 [Branch-and-Bound, 10 Punkte]

Ein Rucksack mit einer Kapazität von **11 Gewichtseinheiten** soll optimal gepackt werden. Die zur Verfügung stehenden fünf Waren dürfen **nur einmal vollständig oder gar nicht** in den Rucksack gepackt werden. Der Wert des Rucksacks ergibt sich aus der Summe der Einzelwerte der eingesetzten Waren.

Sie sollen mittels eines Branch-and-Bound Verfahrens eine zulässige Lösung maximalen Wertes ermitteln.

Ware $i =$	1	2	3	4	5
Gewicht $g_i =$	3	2	10	8	6
Wert $w_i =$	8	4	10	4	2
relativer Wert $\frac{w_i}{g_i} =$	$\frac{8}{3}$	2	1	$\frac{1}{2}$	$\frac{1}{3}$

Knoten des Baumes und Verzweigung: Der zugehörige Baum soll **binär** sein. An jedem Knoten befindet sich eine Menge M_1 bereits fest eingeplanter Waren, und eine Menge M_2 von Waren, die noch für die Lösung zur Auswahl stehen. Der Baum beginnt mit einem Knoten an dem noch keine Ware verplant ist, $M_1 = \emptyset$, und alle Waren noch zur Auswahl stehen, $M_2 = \{1, 2, 3, 4, 5\}$. Für die binäre Verzweigung wird für eine noch zur Auswahl stehende Ware $x \in M_2$ entschieden, ob sie zur Lösung gehört oder nicht zur Lösung gehört, entsprechend wird verzweigt. Die Kinderknoten werden dann mit $M_1 \cup \{x\}$ und $M_2 \backslash \{x\}$ bzw. M_1 und $M_2 \backslash \{x\}$ beschriftet. Bei Gewichtsüberschreitung von $M_1 \cup \{x\}$ (unerlaubte Lösung) wird nur der Zweig M_1 und $M_2 \backslash \{x\}$ angehängt (unär an dieser Stelle!).

An jedem Knoten muss ein $x \in M_2$ der zur Auswahl stehenden Waren für die Verzweigung gewählt werden. Das soll nach folgender Auswahlregel geschehen:

Auswahlregel: Aus der Menge M_2 der noch zur Auswahl stehenden Waren wähle diejenige mit dem größten relativen Wert $\frac{w_i}{g_i}$. Verzweige nach dieser Ware.

Bewertungsfunktion: Plane die noch zur Auswahl stehenden Waren in Reihenfolge abnehmender relativer Werte $\frac{w_i}{g_i}$ (die Größten zuerst!) zusätzlich ein. Falls eine Ware nicht vollständig passt, nimm einen entsprechend großen Anteil davon.

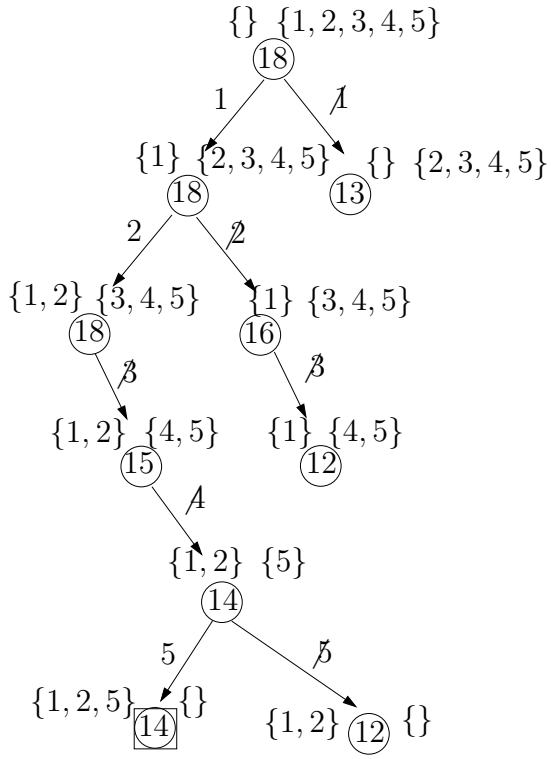
a) Expandieren Sie sukzessive den Baum gemäß Auswahlregel und Bewertungsfunktion, bis die Lösung gefunden wird. Kennzeichnen Sie die Lösung als solche. An jedem Knoten notieren Sie die Menge der bereits eingeplanten Waren, die Menge der noch zur Auswahl stehenden Waren und den Wert der Bewertungsfunktion.

b) Erläutern Sie am Ende der Expansion, warum Sie den Baum nicht weiter expandieren müssen.

Antwort:

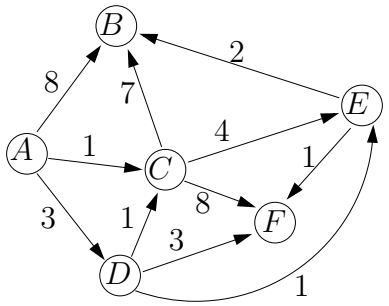
a)

Ware $i =$	1	2	3	4	5
Gewicht $g_i =$	3	2	10	8	6
Wert $w_i =$	8	4	10	4	2
relativer Wert $\frac{w_i}{g_i} =$	$\frac{8}{3}$	2	1	$\frac{1}{2}$	$\frac{1}{3}$



b) Der ausgezeichnete Knoten bezeichnet eine zulässige Lösung mit exaktem Wert 14, diese Schranke ist größer als alle oberen Schranken der nicht expandierten Knoten. Somit kann keine bessere Lösung existieren.

Aufgabe 5 [Kürzeste Wege, 10 Punkte]



a) Wenden Sie den Dijkstra-Algorithmus auf den angegebenen Graphen mit Startknoten A an.

   Geben Sie dabei für jeden Schritt die Knoten der Warteschlange Q und der Welle W nach absteigenden d-Werten in der Form $(x, d(x))$ an.

b) Geben Sie ein möglichst kleines Beispiel dafür an, dass der Dijkstra-Algorithmus mit negativen Kantengewichten auch ohne Zykel nicht korrekt ist. Begründen Sie kurz Ihre Antwort.

Antwort:

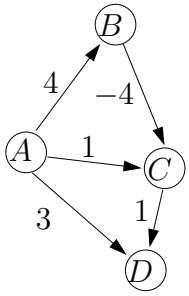
a)

W	Q
$(A,0)$	$(C,1),(D,3),(B,8),(E,\infty),(F,\infty)$
$(A,0),(C,1)$	$(D,3),(E,5),(B,8),(F,9)$
$(A,0),(C,1),(D,3)$	$(E,4),(B,8),(F,6)$
$(A,0),(C,1),(D,3),(E,4)$	$(F,5),(B,6)$
$(A,0),(C,1),(D,3),(E,4),(F,5)$	$(B,6)$
$(A,0),(C,1),(D,3),(E,4),(F,5),(B,6)$	

b) Ein Gegenbeispiel ist in der folgenden Abbildung dargestellt. Nach dem ersten Schritt wird der Knoten C aus der Warteschlange entfernt, der Knoten D bekommt danach den Wert 2 und D wird als nächstes aus W entfernt.

   Falls später der Knoten B an die Reihe kommt, kann zwar der d-Wert von C nochmal auf 0 geändert werden, der Wert von D wird aber nicht mehr angepasst, da C nicht mehr an die Reihe kommt.

   Das Problem ist, dass davon ausgegangen wird, dass für alle Knoten u mit minimalem Wert $d(u)$ gilt, dass $d(u)$ bereits die Länge des kürzesten Weges vom Start zu u angibt.


B
4
−4
A
1
C
3
1
D

Aufgabe 6 [Kontextfreie Sprachen, 10 Punkte]

Sei $G = (V, \Sigma, P, S)$ die kontextfreie Grammatik mit folgenden Produktionen:

$$\begin{aligned} S &\longrightarrow ABCd \\ A &\longrightarrow aA \mid a \\ B &\longrightarrow DEF \mid bcc \\ C &\longrightarrow Cd \mid \varepsilon \\ D &\longrightarrow b \\ E &\longrightarrow DEF \mid \varepsilon \\ F &\longrightarrow cc \end{aligned}$$

a) Geben Sie die Menge der Variablen N und die Menge der Terminalsymbole Σ an.

b) Geben Sie zwei Worte an, die in $L(G)$ liegen.

c) Geben Sie zwei Worte an, die *nicht* in $L(G)$ liegen.

d) **(± 2 Punkte)** Welche der folgenden Aussagen sind wahr, welche falsch?

	Wahr	Falsch
$S \overset{*}{\underset{G}{\Longrightarrow}} \varepsilon$		×
$B \overset{*}{\underset{G}{\Longrightarrow}} \varepsilon$		×
$B \overset{*}{\underset{G}{\Longrightarrow}} bbccc$		×
$B \overset{*}{\underset{G}{\Longrightarrow}} bbcccc$	×	

e) Beschreiben Sie, welche Wörter in L liegen.

f) Ist die Sprache $L(G)$ auch regulär? Begründen Sie Ihre Antwort!

Antwort:

a) $N = \{ S, A, B, C, D, E, F \}$, $\Sigma = \{ a, b, c, d \}$

b) z.B. abccd, aabbccccd, aaabccdd

c) z.B. abc, abcc, bccd

d) s.o.

e) $L = \{ a^n b^m c^{2m} d^k \ ; \ m, n, k > 0 \}$

f) Nein, um die Sprache zu erkennen muß man die Anzahlen der b's und c's zählen, was mit einem DEA nicht möglich ist.

Aufgabe 7 [Primzahltest, 10 Punkte]

Betrachten Sie folgenden Primzahltest:

Eingabe: natürliche Zahl $n > 2$
Ausgabe: true, falls n eine Primzahl ist, false sonst

result := true;
FOR $i = 2$ TO $n - 1$ DO
    IF n durch i teilbar THEN result := false;
END FOR;
RETURN result;

a) Geben Sie die Laufzeit des Algorithmus in O-Notation in Abhängigkeit von n an! Gehen Sie dabei davon aus, dass eine Teilbarkeitsprüfung konstante Zeit benötigt.

b) Begründen Sie, warum der Algorithmus korrekt bleibt, wenn die FOR-Anweisung durch "FOR $i = 2$ TO $\lfloor\sqrt{n}\rfloor$ DO" ersetzt wird. Hierbei steht $\lfloor x \rfloor$ für die größte ganze Zahl z, die $z \leq x$ erfüllt. Was für eine Laufzeit ergibt sich mit dem verbesserten Algorithmus?

c) Sagen die Korrektheit und die Laufzeit des Algorithmus aus b) etwas darüber aus, ob das Problem PRIME in P liegt, oder ob es NP-hart ist? Begründen Sie Ihre Aussage!

Antwort:

a) Da der Teilbarkeitstest $(n-2)$-mal durchgeführt wird, liegt die Laufzeit offensichtlich in $O(n)$.

b) Wenn n eine Primzahl ist, dann findet auch der veränderte Algorithmus keinen Teiler von n. Er gibt also "true" aus, und dies ist korrekt.

Wenn n keine Primzahl ist, dann gibt es einen Teiler $i \in \mathbb{N}$, $1 < i < n$. Das heißt, es gibt eine weitere natürliche Zahl j, so dass $i \cdot j = n$ erfüllt ist. Wegen der Bedingungen an i muss dann auch j die Ungleichungen $1 < j < n$ erfüllen.

Nehmen wir an, dass $i \leq j$ gilt. (Anderenfalls können wir die Rollen von i und j vertauschen.) Dann gilt $i^2 = i \cdot i \leq i \cdot j = n$, also $i \leq \sqrt{n}$. Weil i eine natürliche Zahl ist, folgt $i \leq \lfloor\sqrt{n}\rfloor$. Es gibt also einen Teiler von n, der kleiner gleich $\lfloor\sqrt{n}\rfloor$ ist. Dieser wird vom veränderten Algorithmus gefunden. Er gibt "false" aus. Dies ist korrekt.

Die Laufzeit des veränderten Algorithmus liegt offenbar in $O(\sqrt{n})$.

c) Wäre n die Eingabegröße, dann würde der Algorithmus aus b) zeigen, dass PRIME in P liegt, denn der Algorithmus entscheidet in Zeit $O(\sqrt{n})$, also auch in $O(n)$, ob die gegebene Zahl eine Primzahl ist. Allerdings ist die Eingabegröße in diesem Fall

nicht n, sondern sie liegt in der Größenordnung $\log n$. Die Zahl n hat zum Beispiel in Binärdarstellung $\lfloor \log_2 n \rfloor + 1$ viele Stellen.

Wenn wir also mit k die Eingabegröße bezeichnen, dann liegt die Laufzeit des Algorithmus aus b) im worst-case in $\Theta(\sqrt{n}) = \Theta(n^{\frac{1}{2}}) = \Theta(\exp(k)^{\frac{1}{2}}) = \Theta(\exp(k/2))$. Der Algorithmus hat also eine Laufzeit, die exponentiell in der Größe der Eingabe ist. Daher zeigt seine Laufzeit und Korrektheit nicht, dass PRIME in P liegt. Andererseits kann die Existenz eines Lösungsalgorithmus natürlich auch nichts darüber aussagen, ob PRIME NP-hart ist.

Aufgabe 8 [Greedy-Algorithmen, 10 Punkte]

Eine Baustoffhandlung verleiht kostenlos und unverbindlich einen Betonmischer. Zu Beginn eines Tages liegt eine Liste $S = \{K_1, K_2, \ldots, K_n\}$ interessierter Kunden vor. Zu jedem Kunden K_i ist sowohl die Abholzeit a_i als auch die Rückgabezeit r_i bekannt. Logischerweise gilt $0 \leq a_i < r_i$. Die Aufträge in S seien nach aufsteigender Rückgabezeit sortiert, also gilt $r_1 \leq r_2 \leq \ldots \leq r_n$.

Zwei Kunden K_i und K_j heißen *kompatibel*, falls sie nacheinander bedient werden können, wenn also gilt: $a_i \geq r_j$ oder $a_j \geq r_i$.

Die Baustoffhandlung möchte möglichst viele Kunden zufriedenstellen; gesucht ist also eine größte Teilmenge M von S aus paarweise kompatiblen Kunden.

Bekannt sind folgende Eigenschaften:

(i) Es gibt stets eine optimale Lösung, die den Kunden K_1 enthält.

(ii) Ist M eine optimale Lösung für S mit $K_1 \in M$, so ist $M' := M \backslash \{K_1\}$ eine optimale Lösung für $S_1 := \{\ K \in S\ :\ K$ ist kompatibel zu $K_1\ \}$.

a) Entwerfen Sie einen Greedy-Algorithmus, der auf den Eigenschaften (i) und (ii) beruht.

b) Zeigen Sie Eigenschaft (i).

c) Zeigen Sie Eigenschaft (ii).
Tip: Argumentieren Sie über die Kardinalitäten der Lösungen und betrachten Sie auch eine Lösung M^* zu $S^* := S_1 \cup \{K_1\}$.

Antwort:

a)
$M := \{K_1\}$;
$i := 1$;
FOR $j := 2$ **TO** n **DO**
  IF $a_j \geq r_i$ **THEN**
    $M := M \cup \{K_j\}$;
    $i := j$;
  END IF
END FOR

b) Nehmen wir an, es existiert keine optimale Lösung, die K_1 enthält. Sei M eine optimale Lösung mit $K_1 \notin M$ und $|M| \geq 2$. Enthält M nur einen Kunden, der zu K_1 inkompatibel ist, kann dieser gegen K_1 ausgetauscht werden ohne die Größe von M zu ändern (Widerspruch zur Annahme). Also müsste M zwei (oder mehr) Kunden K_i, K_j enthalten, die nicht zu K_1 kompatibel sind. Für diese gilt

$$a_k < r_1 \wedge a_1 < r_k,\ k \in \{i, j\}\,.$$

Zudem gilt $r_1 \leq r_k$, da die Liste nach Rückgabezeiten sortiert ist, außerdem $a_k < r_k$. Insgesamt gilt also $a_j < r_1 \leq r_i$ und $a_i < r_1 \leq r_j$, insbesondere $a_j < r_i$ und $a_i < r_j$. Die Kunden K_i und K_j sind daher auch untereinander inkompatibel, M kann also keine zwei Kunden enthalten, die nicht zu K_1 kompatibel sind. Dies bedeutet, daß keine optimale Lösung mit $|M| \geq 2$ existieren kann, jede Lösung also nur einen Kunden enthält. Dann ist aber auch $M = \{K_1\}$ eine optimale Lösung.

c) Sei M_1 eine Lösung zu S_1. Zu zeigen ist $|M'| = |M_1|$.

(A): Es muss $|M| \geq |M^*|$ gelten, sonst wäre M keine (maximale) Lösung zu S. Da alle Kunden in S_1 kompatibel zu K_1 sind, gilt $|M^*| = |M_1| + 1$. Nach Definition von M' gilt $|M| = |M'| + 1$. Insgesamt also

$$|M'| + 1 = |M| \geq |M^*| = |M_1| + 1 \quad \Rightarrow \quad |M'| \geq |M_1|\,.$$

(B): Andererseits sind alle Kunden in $S \backslash S_1$ nicht kompatibel zu K_1, können also nicht in M' enthalten sein. M_1 kann also nicht weniger Kunden enthalten als M', also gilt $|M_1| \geq |M'|$. Zusammen mit (A) folgt $|M_1| = |M'|$.