Probeklausur: Lösungsvorschläge

Besprechung in den Übungen

Die Lösungen sind ohne Gewähr – Verbesserungsvorschläge sind willkommen.

Aufgabe 1 Beweisen oder widerlegen Sie folgende Aussagen. ***20 Punkte***

(a) $\binom{n}{2} = \Theta(n^2)$

Lösung:

Die Aussage gilt, denn mit $\binom{n}{2} = \frac{n!}{(n-1)!\cdot 2!} = \frac{n^2-n}{2}$ folgt für $n \geq 3$:

$$\frac{1}{3} \cdot n^2 \leq \binom{n}{2} \leq \frac{1}{2} \cdot n^2$$

(b) $\binom{n}{O(1)} = n^{O(1)}$

Lösung:

Die Aussage gilt. Sei $k \in \mathbb{N}$ eine beliebige Konstante.

$$\binom{n}{k} = \frac{n!}{(n-k)! \cdot k!} = \frac{\prod_{i=n-k+1}^{n} i}{k!} \leq \prod_{i=n-k+1}^{n} i \leq \prod_{i=n-k+1}^{n} n = n^k$$

(c) $\binom{n}{\Omega(1)} = n^{\Omega(1)}$

Lösung:

Die Aussage gilt nicht, da $n \in \Omega(1)$, aber $\binom{n}{n} = 1 = n^0$ und $0 \notin \Omega(1)$.

(d) $o(1) = O(1/n)$.

Lösung:

Die Aussage gilt nicht, da eine Funktion langsamer als $\frac{1}{n}$ gegen 0 gehen kann. Ein Beispiel hierfür ist $1/\sqrt{n}$:

$$\lim_{n\to\infty} \frac{1/\sqrt{n}}{1} = 0 \qquad \Rightarrow 1/\sqrt{n} \in o(1)$$

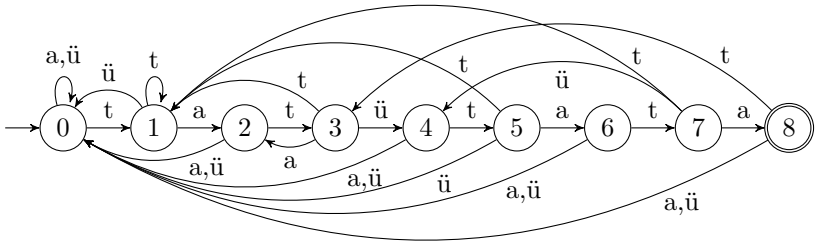
$$\lim_{n\to\infty} \frac{1/\sqrt{n}}{1/n} = \lim_{n\to\infty} \sqrt{n} = \infty \qquad \Rightarrow 1/\sqrt{n} \notin O(1/n)$$

Aufgabe 2 ***32 Punkte***

Gegeben seien das Muster $y =$ tatütata und der Text $x =$ tatatütatatü.

(a) Konstruieren Sie den DFA M_y.

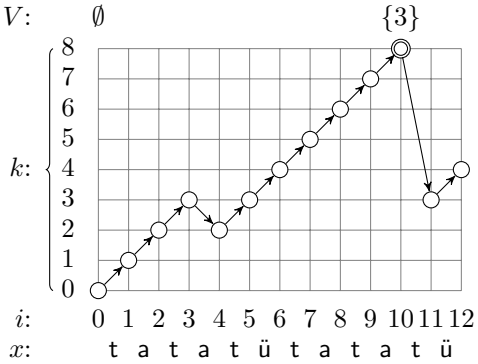
Lösung:



(b) Geben Sie das Ablaufprotokoll von `DFA-String-Matcher`(x, y) an.

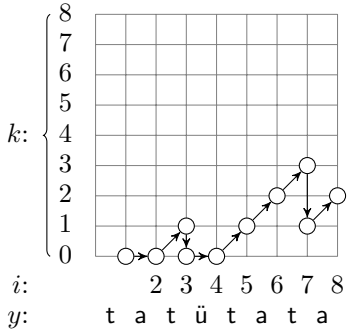
Lösung:

Es ergibt sich folgender Ablauf:



(c) Geben Sie das Ablaufprotokoll von `KMP-Prefix`(y) und die Präfixfunktion π an.

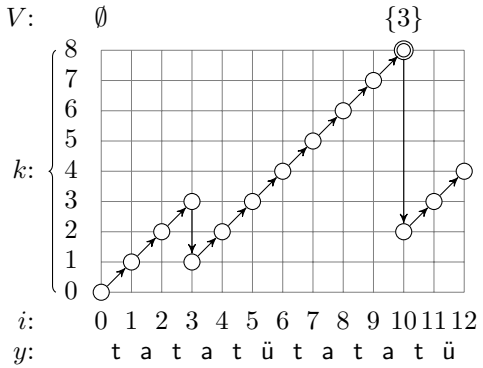
Lösung:



i	0	1	2	3	4	5	6	7	8
y_i		t	a	t	ü	t	a	t	a
$\pi(i)$		0	0	1	0	1	2	3	2

(d) Geben Sie das Ablaufprotokoll von `KMP-String-Matcher`(x, y) an.

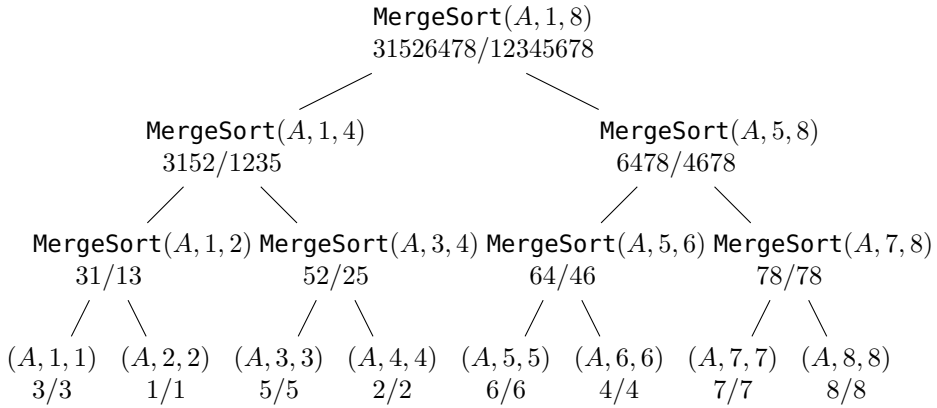
Lösung:



Aufgabe 3 ***10 Punkte***

Geben Sie den Rekursionsbaum von `MergeSort` für das Eingabefeld $A[1,\ldots,8] = (3,1,5,2,6,4,7,8)$ an. Notieren Sie dabei für jeden Aufruf `MergeSort`(A,l,r) von `MergeSort` das Teilfeld $A[l\ldots r]$ zu Beginn und nach Beendigung dieses Aufrufs.

Lösung:



Aufgabe 4 ***24 Punkte***

Gegeben sind n Schrauben der Größe $a_1,\ldots,a_n$ und n dazu passende Muttern der Größe $b_1,\ldots,b_n$, d.h. es gibt eine Permutation π auf $\{1,\ldots,n\}$ mit $a_i = b_{\pi(i)}$ für $i = 1,\ldots,n$. Wir nehmen an, dass nur Vergleiche (a_i,b_j) zwischen Schrauben und Muttern möglich sind und dass die Antwort $a_i < b_j$ oder $a_i = b_j$ oder $a_i > b_j$ ist.

(a) Wie viele Vergleiche sind im besten und im schlechtesten Fall jeweils nötig, um ein Paar (i,j) mit $a_i = b_j$ zu finden? Begründen Sie.

Lösung:

Im besten Fall wird beim ersten Versuch ein passendes Paar gefunden, es genügt ein Vergleich. 0 Vergleiche reichen nicht aus, da sonst mindestens eine der Eingaben $(1,2),(1,2)$ und $(1,2),(2,1)$ falsch behandelt wird.

Im schlechtesten Fall genügen dem folgenden Algorithmus $n-1$ Vergleiche:

```
1 Wähle i ← 1
2 for j ← 1 to n − 1 do
3    if a_i = b_j then return (i, j)
4 return (i, n)
```

Um zu sehen, dass kein Algorithmus im schlechtesten Fall mit weniger Vergleichen auskommt, nehmen wir an, dass ein solcher Algorithmus B existiert und betrachten den Fall, dass in den höchstens $n-2$ gestellten Fragen niemals mit Gleichheit geantwortet wurde. Dann gibt es für jedes a_i noch mindestens 2 b_j, die infrage kommen – davon ist aber (bei paarweise verschiedenen b_j) mindestens eines nicht korrekt.

(b) Geben Sie einen Algorithmus an, der für jede Schraube eine passende Mutter mit durchschnittlich $O(n\log n)$ Vergleichen findet.

Hinweis: Passen Sie `QuickSort` an diese Aufgabenstellung an.

Lösung:

Wähle bei Eingabe (A,B) eine beliebige Schraube $a_p \in A$ (zum Beispiel die letzte) und pivotisiere mit dieser die Muttern: Diese werden dadurch auf drei Teilmengen verteilt – eine mit kleineren $(B_<)$, eine mit gleich großen $(B_=)$ und eine mit größeren $(B_>)$ Muttern als a_p. Verwende nun eine Mutter $b_p \in B_=$, um die Schrauben zu pivotisieren – es ergeben sich $A_<$, $A_=$ und $A_>$. Rufe die Prozedur nun rekursiv für $(A_<,B_<)$ und $(A_>,B_>)$ auf und füge die sortierten Listen aneinander. Danach gilt $a_i = b_i$ für alle $i \in \{1,\ldots,n\}$.

Die Laufzeitabschätzung folgt, da genau doppelt so viele Vergleiche vorgenommen werden wie bei der einfachen Sortierung von A durch `QuickSort`. Der Faktor 2 verschwindet in der $\mathcal{O}$-Notation.

(c) Zeigen Sie, dass hierfür im schlechtesten Fall $\Omega(n\log n)$ Vergleiche nötig sind.

Lösung:

Fixiere eine Schraubenfolge paarweise verschiedener $a_1,\ldots,a_n$. Jeder Algorithmus muss in seiner Berechnung zwischen den $n!$ möglichen Permutationen der b_i unterscheiden, um korrekt antworten zu können. Damit muss der (ternäre) Fragebaum mindestens $n!$ Blätter haben. Mit einer Argumentation analog zu Aufgabe 17 folgt, dass der Fragebaum mindestens $\lceil\log_3(n!)\rceil$ tief sein muss. Mit der Stirling-Formel

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{\mathrm{e}}\right)^n$$

und $\log_3 x = \frac{1}{\log_2 3}\log_2 x$ folgt:

$$\lceil\log_3 n!\rceil \geq \log_3 n! = \Omega\left(\log_3\sqrt{n} + n(\log_3 n - \log_3 \mathrm{e})\right) = \Omega(n\log_2 n)$$

Aufgabe 5 ***22 Punkte***

Sei $G = (V, E)$ ein Graph mit n Knoten, m Kanten und c Zusammenhangskomponenten. Zeigen Sie:

(a) Wenn G ein Wald ist, dann ist $c = n - m$.

Lösung:
Sei G ein Wald. Dann ist jede der c Zusammenhangskomponenten ein Baum. Nenne diese $G_1, \ldots, G_c$, wobei jedes G_i genau n_i Knoten habe. In jeder Komponente gilt der Hinweis, dass $m_i = n_i - 1$. Außerdem gilt $n = \sum_{i=1}^{c} n_i$ und $m = \sum_{i=1}^{c} m_i$. Damit folgt

$$m = \sum_{i=1}^{c}(n_i - 1) = \left(\sum_{i=1}^{c} n_i\right) - c = n - c$$

(b) Wenn $c = n - m$ ist, dann ist G ein Wald.

Lösung:
Sei G ein Graph mit $c = n - m$. Betrachte die c Zusammenhangskomponenten $G_1, \ldots, G_c$ von G. In jeder Komponente G_i muss $m_i \geq n_i - 1$ gelten, da sie sonst nicht zusammenhängend wäre. Da damit bereits alle $m = n - c$ Kanten verteilt sind, muss jeweils $m_i = n_i - 1$ gelten. Nach dem Hinweis sind damit die G_i Bäume, also ist G ein Wald.

(c) $n - c \leq m \leq \binom{n-c+1}{2}$.

Lösung:
Wir zeigen zunächst $n - c \leq m$ durch Induktion über c:

Induktionsanfang: Für $c = 1$ ist G zusammenhängend und damit $m \geq n - 1$.

Induktionsvoraussetzung: In jedem Graph G mit $c - 1$ Zusammenhangskomponenten gilt $m \geq n - (c - 1)$.

Induktionsbehauptung: In jedem Graph G mit c Zusammenhangskomponenten gilt $m \geq n - c$.

Induktionsschluss: Sei G ein Graph mit c Zusammenhangskomponenten. Erzeuge einen Graphen G' aus G durch Verbinden zweier Zusammenhangskomponenten. Es gilt $n' = n$, $m' = m + 1$, $c' = c - 1$; damit ist die Induktionsvoraussetzung anwendbar und es gilt $m' \geq n' - c'$, also $m + 1 \geq n - (c - 1)$ und schließlich $m \geq n - c$.

Nun wenden wir uns der zweiten Ungleichung zu. Hierzu behaupten wir, dass ein Graph mit n Knoten und c Zusammenhangskomponenten genau dann maximal viele Kanten hat, wenn er aus $c - 1$ isolierten Knoten und einer $(n - c + 1)$-Clique besteht – ein solcher Graph hat gerade $\binom{n-c+1}{2}$ Kanten. Angenommen, es gibt in G zwei Zusammenhangskomponenten mit $n_i \geq n_j \geq 2$ Knoten. Dann wählen wir einen Knoten v in der j-ten Komponente und verschieben ihn in die i-te Komponente, indem wir alle zu v inzidenten Kanten löschen (dies sind höchstens $n_j - 1$) und stattdessen n_i Kanten zwischen v und den Knoten in der i-ten Komponente einfügen. Die Anzahl der Knoten und Zusammenhangskomponenten bleibt unverändert, die Anzahl der Kanten wächst um mindestens $n_i - (n_j - 1) \geq 1$ – also hatte G nicht maximal viele Kanten.

Hinweis: G ist genau dann ein Baum, wenn $m = n - 1$ und $c = 1$ ist.

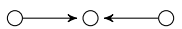
Aufgabe 6 Zeigen oder widerlegen Sie folgende Aussagen. ***12 Punkte***

(a) Ein azyklischer Digraph hat höchstens eine Wurzel.

Lösung:
Die Aussage gilt. Angenommen es gibt einen azyklischen Digraphen G mit zwei unterschiedlichen Wurzeln u, v. Dann sind von u und v aus jeweils alle Knoten erreichbar. Insbesondere gibt es gerichtete Pfade $(u, \ldots, v)$ und $(v, \ldots, u)$. Zusammengesetzt ergeben diese einen Zyklus von u nach u – damit kann G nicht azyklisch sein.

(b) Ein azyklischer Digraph mit n Knoten und $n - 1$ Kanten ist ein gerichteter Baum.

Lösung:
Die Aussage gilt nicht, wie sich an dem folgenden Gegenbeispiel erkennen lässt:

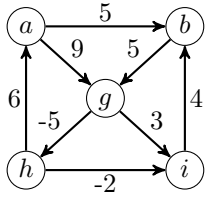


(c) Ein azyklischer Graph mit n Knoten und $n - 1$ Kanten ist ein Baum.

Lösung:
Die Aussage gilt. Angenommen es gäbe einen azyklischen Graphen G mit n Knoten und $n - 1$ Kanten, der unzusammenhängend ist. Dann gibt es mindestens eine Zusammenhangskomponente G_i mit $m_i \geq n_i$, also mindestens so vielen Kanten wie Knoten. Deren Spannbaum hat aber nur $n_i - 1 < m_i$ Kanten und die übrigen $m_i - (n_i - 1) \geq 1$ Kanten schließen jeweils einen Kreis, da zwischen den durch sie verbundenen Knoten bereits ein Pfad über Kanten des Spannbaums existiert. Dies ist ein Widerspruch dazu, dass G azyklisch ist.

Aufgabe 7 ***30 Punkte***

Gegeben sei nebenstehender Digraph G mit Kostenfunktion ℓ.



(a) Bestimmen Sie mit dem Bellman-Ford-Moore-Algorithmus kürzeste Wege von a zu allen anderen Knoten.

(b) Bestimmen Sie mit dem Floyd-Warshall-Algorithmus kürzeste Wege zwischen allen Knoten.

Lösung zu (a):
In der folgenden Tabelle steht jede Spalte für den Inhalt der Warteschlange nach einem Schleifendurchlauf. Die Einträge haben das Format (r, v, p, g), die einem Knoten v

eine Runde r zuordnen und zusätzlich einen neuen Wert p für `parent`(v) und einen neuen Wert g für die Entfernung $g(v)$ von v zum Startknoten a enthalten.

$$\left|\ (0,a,\texttt{nil},0)\ \right|\ \left|\begin{matrix}(1,b,a,5)\\(1,g,a,9)\end{matrix}\right|\ (1,g,a,9)\ \left|\begin{matrix}(2,h,g,\ 4)\\(2,i,g,12)\end{matrix}\right|\ (3,i,h,2)\ \Big|$$

Damit ergibt sich:

v	a	b	g	h	i
$g(v)$ (Entfernung zu a)	0	5	9	4	2
`parent`(v)	`nil`	a	a	g	h

Lösung zu (b):
Die Werte von d_0 ergeben sich aus den Kantenlängen. Die Werte von d_k ergeben sich aus dem Minimum des direkten Wegs und des Umwegs über den k-ten Knoten in d_{k-1} (vgl. Skript). Veränderte Werte sind jeweils fett dargestellt.

d_0	a	b	g	h	i
a	∞	5	9	∞	∞
b	∞	∞	5	∞	∞
g	∞	∞	∞	-5	3
h	6	∞	∞	∞	-2
i	∞	4	∞	∞	∞

d_1	a	b	g	h	i
a	∞	5	9	∞	∞
b	∞	∞	5	∞	∞
g	∞	∞	∞	-5	3
h	6	**11**	**15**	∞	-2
i	∞	4	∞	∞	∞

d_2	a	b	g	h	i
a	∞	5	9	∞	∞
b	∞	∞	5	∞	∞
g	∞	∞	∞	-5	3
h	6	11	15	∞	-2
i	∞	4	**9**	∞	∞

d_3	a	b	g	h	i
a	∞	5	9	**4**	**12**
b	∞	∞	5	**0**	**8**
g	∞	∞	∞	-5	3
h	6	11	15	**10**	-2
i	∞	4	9	**4**	**12**

d_4	a	b	g	h	i
a	**10**	5	9	4	**2**
b	**6**	**11**	5	0	**-2**
g	**1**	**6**	**10**	-5	**-7**
h	6	11	15	10	-2
i	**10**	4	9	4	**2**

d_5	a	b	g	h	i
a	10	5	9	4	2
b	6	**2**	5	0	-2
g	1	**-3**	**2**	-5	-7
h	6	**2**	**7**	**2**	-2
i	10	4	9	4	2

In $d_5(i,j)$ ist der kürzeste Weg von i nach j gespeichert.

Aufgabe 8 ***20 Punkte***
Zur Berechnung der transitiven Hülle eines als Adjazenzmatrix A gegebenen Digraphen wurde folgender Algorithmus vorgeschlagen:

Prozedur `Finde-Wege`$(A[1\ldots n, 1\ldots n])$

```
1  for i := 1 to n do
2    for k := 1 to n do
3      if A[i,k] = 1 then
4        for j := 1 to n do
5          if A[k,j] = 1 then A[i,j] := 1
```

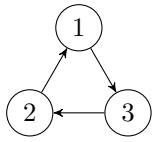
(a) Bestimmen Sie die asymptotische Laufzeit im besten und im schlechtesten Fall.

Lösung:
Im besten Fall sind alle Einträge der Matrix 0 und die innere Schleife wird nie ausgeführt – die Laufzeit ist dann $\Theta(n^2)$ wegen der beiden äußeren Schleifen.
Im schlechtesten Fall sind alle Einträge 1, sodass die innere Schleife stets ausgeführt wird – dann ist die Laufzeit $\Theta(n^3)$ wegen der drei `for`-Schleifen.

(b) Zeigen Sie, dass die Prozedur zwar alle Wege der Länge 2, aber nicht alle Wege der Länge 3 findet.

Lösung:
Sei (u,v,w) ein Weg der Länge 2. Bei $i=u$ und $k=v$ ist die Bedingung $A[i,k]=1$ erfüllt und die innere Schleife wird ausgeführt. Wenn dort $j=w$ gilt, ist $A[k,j]=1$ und die »Abkürzung« (u,w) wird durch $A[i,j]:=1$ gespeichert.
Es werden aber nicht alle Wege der Länge 3 gefunden, wie sich an folgendem Beispiel nachvollziehen lässt:

A	1	2	3
1	0	0	1
2	1	0	0
3	0	1	0



Die Kante $(1,2)$ wird erst bei $i=1$ und $k=3$ gefunden – und dann ist es zu spät, $(1,1)$ zu finden, da dafür $k=2$ und $i=1$ sein müsste.

(c) Modifizieren Sie die Prozedur so, dass sie alle Wege der Länge ≥ 1 in Zeit $\mathcal{O}(n^3)$ findet. (*Hinweis:* Orientieren Sie sich an dem Floyd-Warshall-Algorithmus.)

Lösung:
Es muss die Schleife über k nach außen gezogen werden:

Prozedur `Finde-Alle-Wege`$(A[1\ldots n, 1\ldots n])$

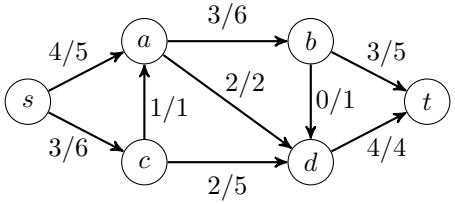
```
1  for k := 1 to n do
2    for i := 1 to n do
3      if A[i,k] = 1 then
4        for j := 1 to n do
5          if A[k,j] = 1 then A[i,j] := 1
```

Dann gilt für die äußere Schleife folgende Invariante:

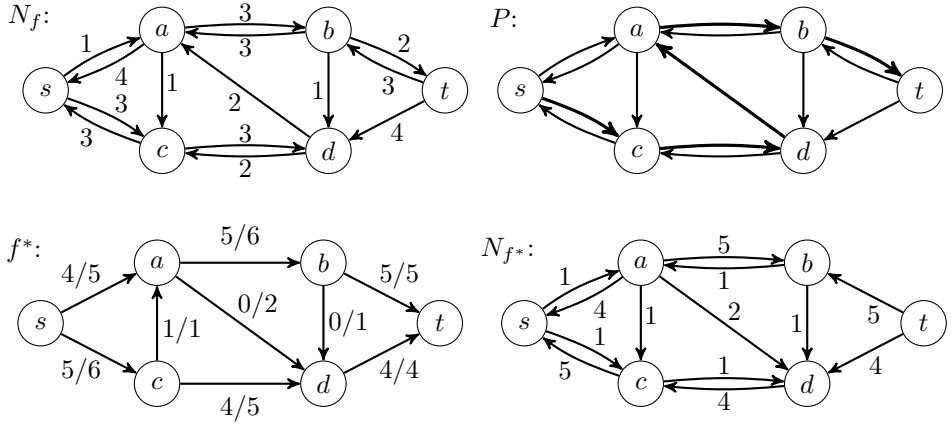
$$\forall i,j : (\exists \text{ ein } (i,j)\text{-Pfad mit inneren Knoten } v \leq k) \Leftrightarrow A[i,j] = 1$$

Aufgabe 9 ***20 Punkte***

(a) Bestimmen Sie mit dem Ford-Fulkerson-Algorithmus ausgehend von nebenstehendem Fluss f einen maximalen Fluss f^* von s nach t. Geben Sie insbesondere für jeden Schleifendurchlauf das Restnetzwerk und den Zunahmepfad an.



Lösung:
Aus f ergibt sich das Restnetzwerk N_f. Bei Wahl des Zunahmepfads P (der zugehörige Fluss f_P hat die Kapazität $c_f(P) = 2$) ergibt sich der Fluss $f^* := f + f_P$. Dieser ist maximal und der Algorithmus bricht ab, da in N_{f^*} kein Pfad von s nach t existiert.



(b) Geben Sie einen Schnitt S minimaler Kapazität an (mit Begründung).

Lösung:
Wähle $S := \{s, a, b, c, d\}$ und $T := \{t\}$ mit $c(S, T) = 9$. Da S die Menge aller im Restnetzwerk N_{f^*} des maximalen Flusses f^* von s aus erreichbaren Knoten ist, ist dieser Schnitt nach dem Min-Cut-Max-Flow-Theorem minimal – denn wenn ein Schnitt mit kleinerer Kapazität existieren würde, könnte f^* diesen nicht passieren.

Als Bearbeitungszeit sind 190 min (also 1 min pro Punkt) vorgesehen.