13. Übung mit Musterlösung

Lösung 1

Teil 1.Multiple Choice)

Bewertung: Ein Punkt für richtige Antwort, für jede falsche Antwort ein Punktabzug.

a) Für die Exponentialverteilung ist die Zeit bis zum nächsten Ereignis unabhängig davon, wann das letzte Ereignis stattgefunden hat.
JA - Markov-Eigenschaft der Exponentialverteilung (Kap 4, p 92)

b) Im allgemeinen gilt: ein Deadlock tritt auf gdw. der Resource-Allocation Graph einen Zykel enthält.
JA - Zyklus im Graph $\Leftrightarrow$ Deadlock (Kap 6, p 217)

c) Das Round-Robin Verfahren gehört zu den nicht-preemptiven Strategien.
NEIN - Preemptiv, Prozesse werden nach Ablauf des Zeitquantums unterbrochen.

d) Der Banker's Algorithmus wird zur Synchronisation von Prozessen eingesetzt.
NEIN - Der Banker's Algorithmus wird zur Deadlock Vermeidung eingesetzt.

e) Die Exponentialverteilung approximiert die Poissonverteilung.
NEIN

f) Die Belady Anomalie kann beim FIFO Algorithmus auftreten.
JA (Kap 8, p. 361)

g) RMS und EDF sind beides preemptive Realtime-Scheduling Strategien.
JA (Exkurs Realtime-Scheduling)

h) Deadlocks werden in Linux und Windows unterschiedlich behandelt.
NEIN (Exkurs Linux Windows, p. 41)

Lösung 2

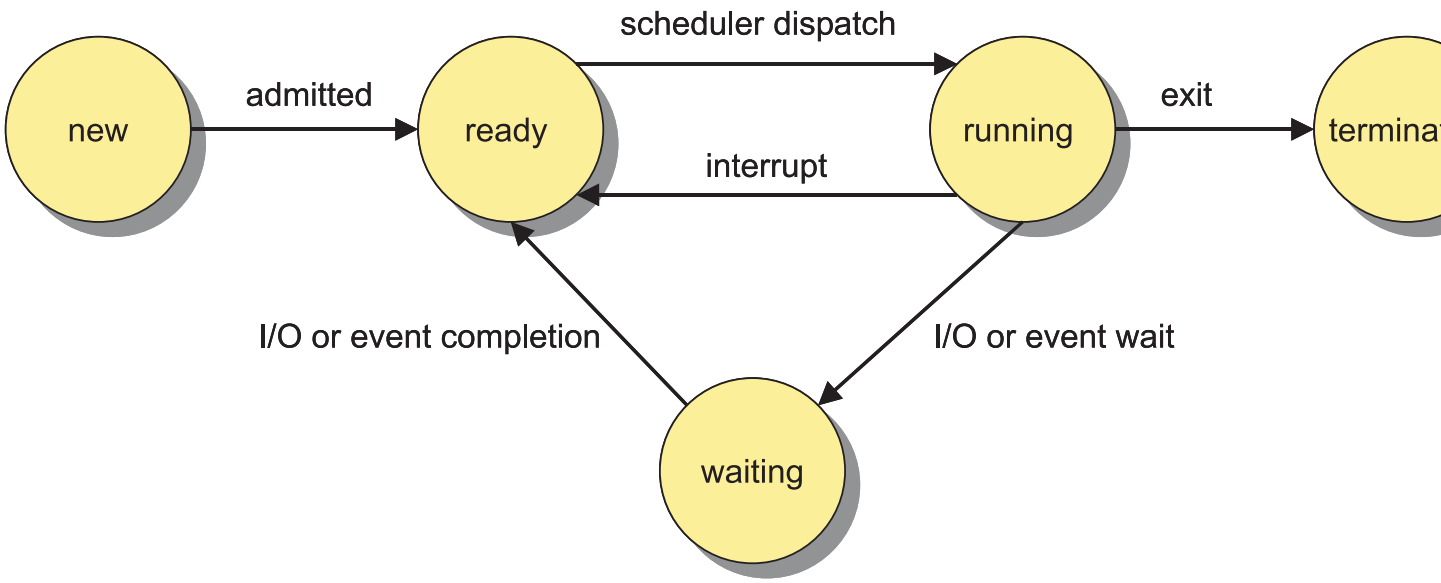
Teil 2.Prozess-Management)

a) Definieren Sie den Begriff Prozess. In welchen Zuständen kann sich typischerweise ein Prozess befinden? Stellen Sie die Zustände in einem Graphen dar, aus dem ersichtlich wird, von welchem Prozesszustand in welchen anderen Prozesszustand gewechselt werden kann.

Prozeß :=

Programm in Ausführung zu einem bestimmten Zeitpunkt, d.h. ein Prozeß umfaßt den Programm-Code (=Textsegment), den aktuellen Wert des PC (program counters) und der anderen CPU-Register und den Wert aller Programm-Variablen (inkl. Files).

- **new**: Ein neuer Prozeß wird gerade erzeugt.
- **ready**: Prozeß ist bereit zur Ausf¡FC¿hrung und wartet auf die Zuteilung der CPU.
- **running**: Prozeß wird gerade ausgeführt (d.h. verfügt über die CPU)
- **waiting**: Prozeß wartet auf ein (eigenes) Betriebsmittel oder I/O-Ereignis
- **terminated**: Prozeß wurde vollst¡E4¿ndig ausgef¡FC¿hrt und beendet.
- blocked (statt 'waiting'): Prozeß wartet auf ein Betriebsmittel, das von einem anderen Proze¡DF¿ belegt ist.
- killed (statt 'terminated'): Prozeß wurde vorzeitig, z.B. durch ein entsprechendes Signal, beendet.



Graphik 1.5 Punkt, Definitionen je 0.5 Punkte, Prozess 1 Punkt, insgesamt 5 Punkte

b) Definieren Sie den Begriff Thread. Welche Vorteile bietet die Nutzung mehrer Threads im Gegensatz zur Nutzung mehrer Prozesse?
Thread: lightweight process, kein eigenes Text-und Datensegment. Vorteile: Threads teilen sich den gleichen Adressraum, kein Overhead beim Thread-Wechsel (Kap. 2, p. 41)
Punkte: Definition 1, Vorteile: je 0.5

c) Mit welchem Kommando kann man unter Unix Prozesse abbrechen? Mit welchem Kommando kann man sich die aktiven Prozesse darstellen lassen?
`kill -9`
`ps a` / `ps -A`
Punkte: jeweils einen Punkt, insgesamt 2 Punkte, Falls `top` statt `ps` genannt wurde 0.5P

d) Durch welche zwei Verfahren kann Interprozesskommunikation erfolgen? Erläutern Sie kurz die beiden Verfahren.

- **Message Passing**:
  Über das Betriebssystem wird eine Verbindung zwischen P1 und P2 aufgebaut.
  System-Calls: getHostID, open/accept/close connection.

  – Vorteil: hohe Flexibilität (Zahl der Prozesse, Länge der Nachricht, ...)
  – Nachteil: langsam (wegen OS)

- **Shared Memory**:
  Unabhängig vom OS haben P1 und P2 Zugriff auf einen gemeinsamen Speicherbereich.
  – Vorteil: schnell
  – Nachteil: Synchronisation erforderlich

insgesamt 3 Punkte, jeweils 1.5

Lösung 3

Teil 3.Prozess-Scheduling)

Fünf Prozesse $P_1, ..., P_5$ mit den Laufzeiten $L_1 = 2, L_2 = 7, L_3 = 3, L_4 = 6, L_5 = 5$. Diese Prozesse sollen mittels der Verfahren FCFS, SJF, Round-Robin mit Zeitquantum vier (RR4) auf zwei CPUs A und B verteilt werden.

a)

		Zeitpunkte													
Verfahren	CPU	1	2	3	4	5	6	7	8	9	10	11	12	13	14
FCFS	A	1	1	3	3	3	4	4	4	4	4	4			
	B	2	2	2	2	2	2	2	5	5	5	5	5		
SJF	A	1	1	5	5	5	5	5	2	2	2	2	2	2	2
	B	3	3	3	4	4	4	4	4	4					
RR4	A	1	1	3	3	3	5	5	5	5	4	4	5		
	B	2	2	2	2	4	4	4	4	2	2	2			

Punkte: 1P FCFS, 1P SJF, 2P RR4

b) Berechnen Sie für jedes der drei Verfahren die mittlere Wartezeit!

a) $2+5+7 = 14 \quad \Rightarrow 14/5 = 2.8$

b) $2+7+3 = 12 \quad \Rightarrow 12/5 = 2.4$

c) $0+(0+4)+2+(4+1)+(5+2) = 18 \quad \Rightarrow 18/5 = 3.6$

Punkte: 1P pro korrekter Wartezeit

Lösung 4

Teil 4.Wechselseitiger Ausschluss)

a) Der Algorithmus `scheduler` verteilt die eintreffenden Druckjobs vom Cluster auf die Drucker. Markieren Sie die Zeilen, die den Algorithmus in Pseudocode beschreiben.

√	`init (S,1);`	☐	`init (job,1);`
☐	`scheduler(PrinterJob S) {`	√	`scheduler(PrinterJob job) {`
√	`wait(S);`	☐	`signal(S);`
☐	`if(q1.size() >= 4) then {`	√	`if(q1.size() < 4) then {`
☐	`job=q1.dequeue(); }`	√	`q1.enqueue(&job); }`
√	`else { if(q2.size() < 10) then {`	☐	`if(q2.size() == 10) then {`
☐	`q1.enqueue(&job); }`	√	`q2.enqueue(&job); }`
☐	`else { job=q3.dequeue(); }`	√	`else { q3.enqueue(&job); } }`
√	`signal(S); }`	☐	`wait(S); }`

b) Der Algorithmus `drucker1` verarbeitet die Jobs des ersten Druckers. Markieren Sie die Zeilen, die den Algorithmus in Pseudocode beschreiben.

√	`drucker1 {`	☐	`drucker1(job.dequeue()) {`
☐	`PrinterJob S;`	√	`PrinterJob *job;`
☐	`if(q1.size()==0) then {`	√	`repeat {`
√	`job=NULL;`	☐	`job=q1.enqueue();`
√	`wait(S);`	☐	`q1.dequeue();`
☐	`wait(S);`	√	`if(q1.size()>0) then {`
☐	`q1.enqueue(&job);`	√	`job=q1.dequeue(); }`
☐	`if(q1.size()==0) then {`	√	`signal(S);`
√	`if(job!=NULL) then {`	☐	`job=q1.dequeue();`
☐	`signal(S); }`	√	`PrintIt(job); }`
√	`until false; }`	☐	`wait(S); }`

Lösung 5

Teil 5.Deadlocks)

a) Prüfen Sie für die Zustände A und B, ob sich das System in einem sicheren Zustand befindet!

A ist unsicher:

	ALLOC			NEED			WORK			
0	0	2	2	7	3	1	2	1	1	(P_3)
1	3	1	1	3	1	1	3	2	1	(P_1)
2	3	2	1	6	5	0	6	3	2	-
3	1	1	0	2	1	0				
TOTAL	7	6	4							
FREE	2	1	1							

Punkte: 1P für NEED, 1P für WORK
falls NEED fehlt: korrektes Ergebnis 2P, falsches Ergebniss 0P

B ist sicher:

	ALLOC			NEED			WORK			
0	1	2	1	6	3	2	2	1	2	(P_3)
1	3	1	2	3	1	0	3	2	2	(P_1)
2	2	2	0	7	5	1	6	3	4	(P_2)
3	1	1	0	2	1	0	7	5	5	(P_0)
TOTAL	7	6	3				9	7	5	ok
FREE	2	1	2							

Punkte: 1P für NEED, 1P für WORK
falls NEED fehlt: korrektes Ergebnis 2P, falsches Ergebniss 0P

b) 1.Forderung: Request(3)=(0,2,0) : nicht zulaessig. Die Anforderung geht ueber MAX(3) hinaus.

2.Forderung: Request(2)=(2,0,0).

	ALLOC			NEED			WORK			
0	0	2	0	7	3	3	2	1	3	(P_3)
1	2	2	1	4	0	1	3	2	3	-
2	4	1	1	5	6	0	8	4	3	
3	1	1	0	2	1	0	8	6	3	
TOTAL	7	6	2				9	7	4	
FREE	2	1	3							

$\Rightarrow$ nicht zulaessig

3.Forderung: Request(1)=(2,0,0)

	ALLOC			NEED			WORK			
0	0	2	0	7	3	3	2	1	3	(P_3)
1	4	2	1	2	0	1	3	2	3	(P_1)
2	2	1	1	7	6	0	7	4	4	(P_0)
3	1	1	0	2	1	0	7	6	4	(P_2)
TOTAL	7	6	2				9	7	5	ok
FREE	2	1	3							

$\Rightarrow$ zulaessig

Punkte: ohne (korrekte) Begruendung: 0 Punkte
1.Forderung: 1P; 2. und 3.Forderung zusammen 3P (je: 0.5P fuer Alloc/Need, 1P fuer Work) (insg. 4P)

Lösung 6

Teil 6.Speicherverwaltung)

a) Was versteht man unter virtuellem Speicher?

Der virtuelle Speicher bezeichnet den vom tatsächlich vorhandenen Arbeitsspeicher unabhängigen Adressraum, der einem Prozess für Daten und das Programm vom Betriebssystem zur Verfügung gestellt wird.

b)

$\omega =$	3	1	4	0	2	1	3	4	0	2	2	1	4	5	6	2	1	3	4	2	3
Seitenfehler				$\star$																	
Seitenrahmen 1	3	1	4	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Seitenrahmen 2		3	1	4	4	4	4	4	4	4	4	4	4	5	6	6	6	3	3	3	3
Seitenrahmen 3			3	1	1	1	3	3	0	0	0	1	1	1	1	1	1	1	4	4	4

Seitenfehler																					
Seitenrahmen 1	3	1	4	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Seitenrahmen 2		3	1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Seitenrahmen 3			3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Seitenrahmen 4				3	3	3	3	0	0	0	0	0	0	5	6	6	6	3	3	3	3

Lösung 7

a) Was ist der Unterschied zwischen *externer* und *interner* Fragmentierung? externe Fragmentierung: Speicherlücken, die beim Segmenting auftreten (Verschnitt durch unterschiedliche Segmentlänge n).
interne Fragmentierung: Speicherlücken, die beim Paging auftreten (Verschnitt innerhalb der Seite). Jeweils einen Punkt (insgesamt 2)

b) Wieviele Speicherworte stehen dem Programm im physikalischen Speicher zur Verfügung?
975 1 Punkt

c) Welches ist die kleinste und welches die größte für das Programm verfügbare physikalische Adresse?
1710, 4354 Jeweils einen Punkt (insgesamt 2)

d) Berechnen Sie zu den folgenden physikalischen Adressen jeweils die logischen Adressen. Welche Anfragen lösen einen Segmentation Fault aus?

1. 4270 segfault
2. 1810 100
3. 2888 segfault
4. 1935 205
5. 2777 445
6. 4222 872

Jeweils einen halben Punkt.

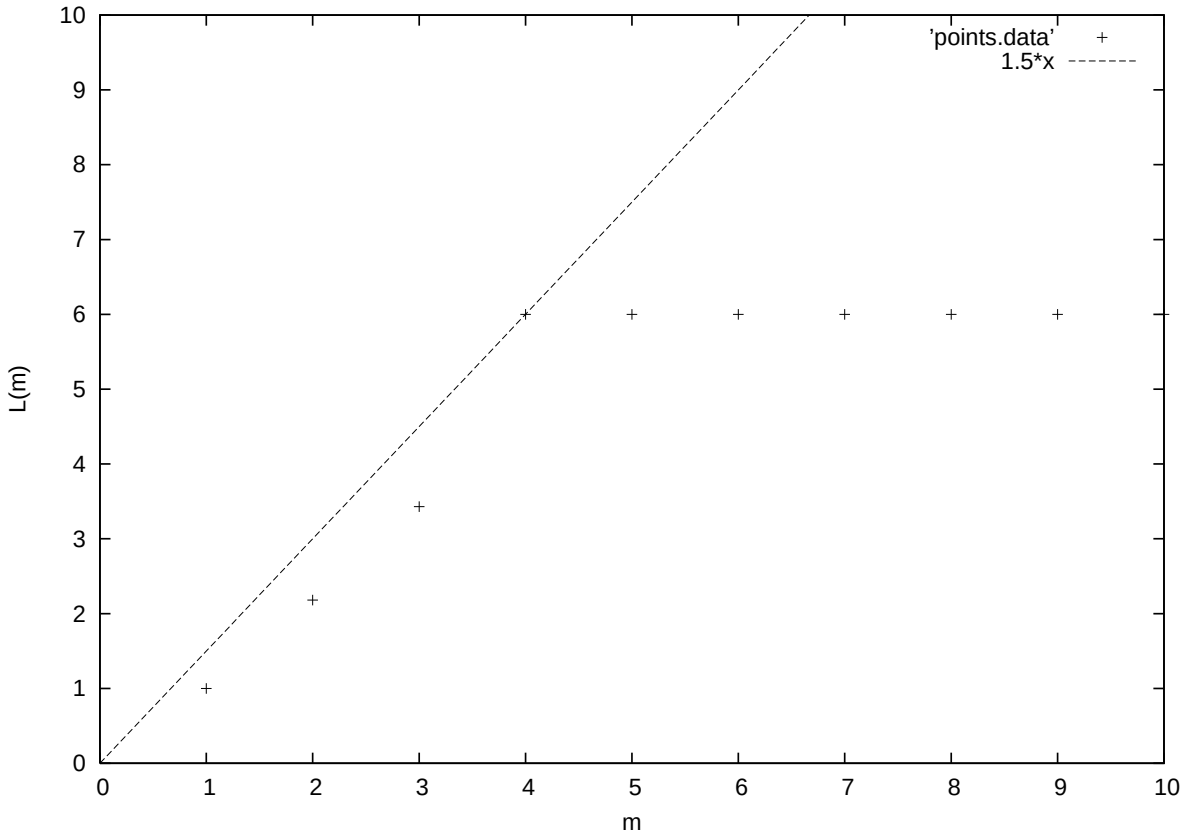
Lösung 8

Teil 8.a))

Bewertung: $m = 10..4, 3, 2, 1$ jeweils 1 Punkt, falls richtig.

ω	2	3	2	3	1	3	1	3	0	3	0	1	2	3	2	3	2	1	2	0	2	1	2	0
$m = 10..4$	**2**	**3**	2	3	**1**	3	1	3	**0**	3	0	1	2	3	2	3	2	1	2	0	2	1	2	0
		2	3	2	3	1	3	1	3	0	3	0	1	2	3	2	3	2	1	2	0	2	1	2
					2	2	2	2	1	1	1	3	0	1	1	1	1	3	3	1	1	0	0	1
									2	2	2	2	3	0	0	0	0	0	0	3	3	3	3	3
$m = 3$	**2**	**3**	2	3	**1**	3	1	3	**0**	3	0	1	**2**	**3**	2	3	2	1	2	**0**	2	1	2	0
		2	3	2	3	1	3	1	3	0	3	0	1	2	3	2	3	2	1	2	0	2	1	2
					2	2	2	2	1	1	1	3	0	1	1	1	1	3	3	1	1	0	0	1
$m = 2$	**2**	**3**	2	3	**1**	3	1	3	**0**	3	0	**1**	**2**	**3**	2	3	2	**1**	2	**0**	2	**1**	2	**0**
		2	3	2	3	1	3	1	3	0	3	0	1	2	3	2	3	2	1	2	0	2	1	2
$m = 1$	2	3	2	3	1	3	1	3	0	3	0	1	2	3	2	3	2	1	2	0	2	1	2	0

m	1	2	3	4	5	6	7	8	9	10
$L(m)$	1	2.18 (24/11)	3.43 (24/7)	6	6	6	6	6	6	6



Teil 8.b))

Bewertung: optimale Speichergröße: 1 Punkt. Gerade richtig eingezeichnet: 1 Punkt.

$m_{opt} = 4$. Die zugehörige Gerade bei Anwendung des Knie-Kriteriums (s.o).

Lösung 9

Teil 9.Working Set)

Es sei der folgende Referenzstring eines Prozesses gegeben:

$$\omega = 2\ 1\ 5\ 0\ 3\ 4\ 2\ 3\ 2\ 2\ 5\ 6\ 1\ 2\ 3\ 4\ 5\ 6\ 0\ 1\ 6\ 5\ 6\ 1$$

Der Wert h soll für diese Aufgabe $h := 5$ betragen.

a) Geben Sie die Definition des Working Sets $W(t,h)$ zum Zeitpunkt t mit Rückwärtsfenster h an.
Der Working-Set $W(t,h)$ diesen Prozesses wird definiert als die Menge der Pages:

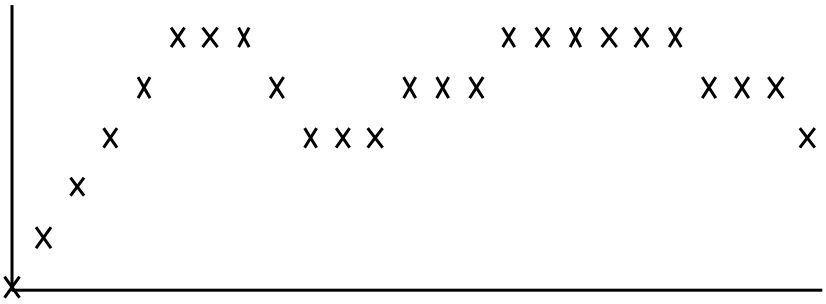
$$W(t,h) := \bigcup_{\tau=t-h+1}^{t} \{r_\tau\}$$

mit Speicher-Referenzstring

$$r_1^T = r_1, \ldots, r_t, r_{t+1}, \ldots, r_T$$

Formel $W(t,h)$ 2 Punkte, Speicherstring 1 P. Insgesamt 3 Punkte

b) Konstruieren Sie einen Graphen:



Achsenbeschriftung 1 Punkt, falsche Zuordnungen -0.5, insgesamt 4 Punkte

c) Geben Sie die Working Sets der lokalen Bereiche zu den Zeitpunkten 10, 15, 19 und 24 an.

10: 2,3,4

15: 1,2,3,5,6

19: 0,3,4,5,6

24: 1,5,6

Jeweils halber Punkt, insgesamt 2 Punkte

Lösung 10

Teil 10.UNIX Inodes)

Pointertyp	Anzahl direct pointer	Referenzierte Daten [Bytes]		
direct	12	$12 \cdot 2K =$	24	K
indirect	1024	$1024 \cdot 2K =$	2	M
double indirect	1024^2	$1024^2 \cdot 2K =$	2	G
triple indirect	1024^3	$1024^3 \cdot 2K =$	2	T

Punkte: je 1P

Lösung 11

Teil 11.Stack Algorithmus) Inklusionseigenschaft:

Analog zur Vorlesung: $M_t(m) \subseteq M_t(m+1), \qquad m = 1,\ldots,n-1$

t	1	2	3	4	5	6	7	8	9
r_t	6	5	4	3	2	5	6	3	1
$M_t(m=6)$	6	5	4	3	2	2	2	2	1
	-	6	5	4	3	3	3	3	2
	-	-	6	5	4	4	4	4	3
	-	-	-	6	5	5	5	5	4
	-	-	-	-	6	6	6	6	5
	-	-	-	-	-	-	-	-	6
$M_t(m=5)$	6	5	4	3	2	2	2	2	1
	-	6	5	4	3	3	3	3	3
	-	-	6	5	4	4	4	4	4
	-	-	-	6	5	5	5	5	5
	-	-	-	-	6	6	6	6	6
$M_t(m=4)$	6	5	4	3	2	2	2	3	1
	-	6	5	4	4	4	4	4	4
	-	-	6	5	5	5	5	5	5
	-	-	-	6	6	6	6	6	6
$M_t(m=3)$	6	5	4	3	2	2	2	3	1
	-	6	5	5	5	5	5	5	5
	-	-	6	6	6	6	6	6	6
$M_t(m=2)$	6	5	4	3	2	5	5	3	1
	-	6	6	6	6	6	6	6	6
$M_t(m=1)$	6	5	4	3	2	5	6	3	1

man sieht hier leicht, dass die Inklusionseigenschaft erfüllt ist. Als nächstes folgt die Herleitung der Prioritätsliste π_t mit dem Kriterium "wachsende Zeit des Hauptspeichereintritts" und daraus die Herleitung des Stacks s_t. Der Stack kann auch direkt ohne die Prioritätsliste aus den $M_t(m)$ hergeleitet werden:

t	1	2	3	4	5	6	7	8	9
r_t	6	5	4	3	2	5	6	3	1
π_t	6	6	6	6	6	6	6	6	6
	-	5	5	5	5	5	5	5	5
		-	4	4	4	4	4	4	4
			-	3	3	3	3	3	3
				-	2	2	2	2	2
					-	-	-	-	1
s_t	6	5	4	3	2	5	6	3	1
	-	6	6	6	6	6	5	6	6
		-	5	5	5	2	2	5	5
			-	4	4	4	4	4	4
			-	-	3	3	3	2	3
			-	-	-	-	-	-	2

Für s_t muss gelten: $M_t(m) = s_t(1), \ldots, s_t(m), \qquad m = 1, \ldots, 6 \ \forall t$